DECLOUD: AN UNSUPERVISED DECONVOLUTION TOOL FOR BUILDING GENE EXPRESSION PROFILES

Submitted by

Øystein Kjørsvik

Department of Informatics

In partial fulfillment of the requirements

For the Degree of Master of Informatics

University of Bergen

Bergen, Norway

Spring 2018

Abstract

Deconvolution is the process of decomposing a mixed signal into its originating elements. For my thesis I created a clustering application, named DeCloud, with the intent to replace the unsupervised clustering step in the deconvolution tool, Deblender. Utilizing clustering packages in R such as optCluster, the application was built to allow for a range of new clustering algorithms. In this thesis the scope has been set to test Hierarchical clustering and two variations of PAM. A novel filtering function was created, providing a different approach to handling clusters. The novel approach has been implemented for use with the PAM clustering method, but could be applied to other algorithms as well. We have tested the resulting pipeline on the data sets used to benchmark Deblender and other tools. Comparing the results obtained by Deblender and by DeCloud, shows that DeCloud obtains marked better results on two of the three datasets used for testing. The last dataset is a complicated case, none of the applications are able to effectively cluster and deconvolve. The novel filter function applied to the PAM algorithm has been shown to be the best performer in each of the two successful deconvolution datasets.

Acknowledgement

I would first like to thank my thesis advisor Inge Jonassen of the Bioinformatics department at University of Bergen and Konstantina Dimitrakopoulou, the creator of Deblender. Konstantina has been a terrific resource at breaking down the deconvolution process in Deblender, helping me logically navigate through the difficulties I have had when creating my application DeCloud. Inge has been very helpful and patient with my thesis draft, it has been a long process. I also want thank my grandparents for the long hours they spent helping me proofread the thesis, their attention to details is remarkable and has been invaluable to me at the end of this process.

INTRODUCTION

Aims of study

The goal of this study was to explore alternative algorithms for the unsupervised clustering step in the deconvolution tool Deblender and to see if this could improve the result of the whole analysis. The unsupervised step in Deblender is currently limited to two different clustering algorithms, K-means and K-mediods. By expanding on the clustering algorithm library we were hoping to see improved results and increase the usability of the application. A secondary goal was to explore possible ways for post-processing of clustering results and to use this to further improve deconvolution results.

Gene Expression

Gene expression is vital in biology to understand life and how organisms differ. The human genome encodes approximately 25 000 genes. The biological differences we observe in the population around us are based on how these genes are expressed. It is estimated that ~83% of human genes are expressed differentially amongst individuals and ~17% amongst populations [42]. Understanding how genes are expressed both in individuals and in larger populations can be vital in how science approaches disease in the future [1, 40].

All humans are fundamentally created from 4 unique essential bases, “A, T, G, C”, these bases are repeated in different patterns over and over again. We know these repeating patterns as DNA. Short strings of DNA make up our genes, what these genes do is determined by what combination of “A, T, G, C” the string is made up by. One of the functions of genes is to tell the body how to build specific proteins, the building blocks of our bodies. Every cell in the human body contains proteins. Another important gene function is the creation of noncoding RNA’s,

which assist the body in many essential functions. We can obtain clues to how a gene functions based on where a gene is expressed in the cell or in the entire organism. Understanding and monitoring gene expression can give us much understanding of what the body is doing and why [2, 30-32, 43].

Gene regulation, as part of gene expression, is how each cell expresses a subset of its genes. Each cell expresses a specific set of genes. These sets depend on the type of cell, the condition of the cell and the surrounding signal. Each cell has different regulatory mechanisms which decide what genes are expressed. These patterns within genes allows for major differences and functions, such as if the cell would become a brain cell or muscle cell [3].

The first level of gene regulation happens during the initiation of transcription, the start of the protein production process. For cells to be adaptable however, they also have downstream processes where genes are regulated again during translation which are often subject to environmental factors. In eukaryotes transcription factors may be very complex and need multiple proteins in the process, compared to the usual single protein in prokaryotes [1, 4].

Microarray

Microarray is a tool to measure gene expression, one type has a glass slide where DNA molecules are attached in certain spots which each is tied to a known gene or DNA sequence. One method for performing a microarray analysis is to use two samples of mRNA, one experimental sample and one reference sample. The samples could for example be from a healthy individual as a baseline (reference) and one sample from an individual with a disease (experimental). The two different mRNA samples are then turned into complementary DNA, also

known as cDNA. These cDNA samples are then labelled with separate fluorescent dyes, often a red dye for one and green dye for the other. This is necessary for the comparison process after hybridization. Hybridization is the process of mixing the samples together and letting the cDNA bind itself to the microarray glass slide where the molecules will bind itself to the spots with complimentary probes. After this, the glass slide is washed and the molecules which are strongly bound to the probes will remain. After the hybridization process, cDNA now has a glass slide with two samples, each sample with its own dye color. To measure which genes are more expressed in each sample, or equally expressed, one just needs to evaluate the prevalent color in each spot. To measure this, a laser is used to target the spots on the hybridized microarray and return a color on each spot. Conceptually, if a spot is red then the red sample is more expressed than the green one. The green dye come through if the opposite true. If the spot is yellow then that gene is equally expressed in both samples. If nothing is bound on a location, then the location on the microarray will appear as black. This allows for easy analysis of difference in gene expression between two different samples [5-6].

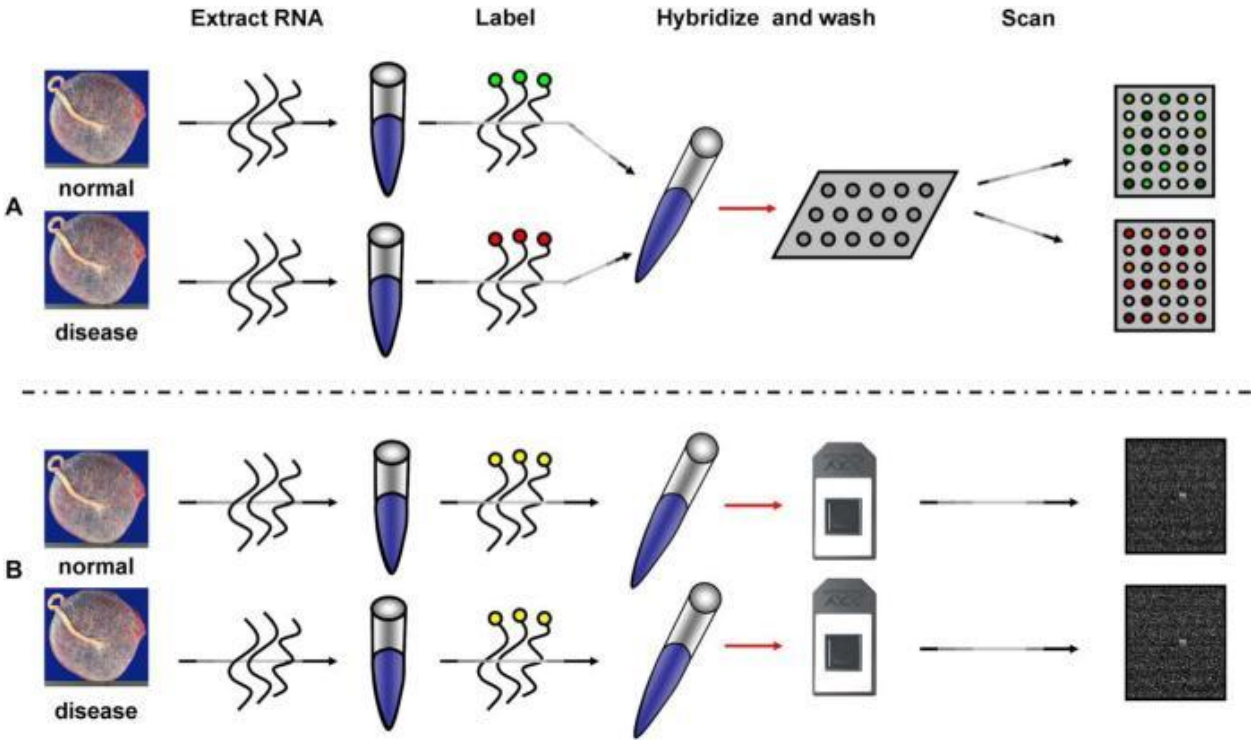

Figure.1) The upper panel, A, representing the two-channel microarray process. Lower panel, B, representing the single-channel microarray [49].

A newer option to measuring gene expression is single-channel microarray. This method has the same initial steps as two-channel microarray, but does not give separate labels to the samples or combine the samples for hybridization. For the singe-channel microarray method, the samples are handled individually (not hybridized together), using the same dye color.

In order to extract data from the single-channel microarray experiment we can analyze its output. The output given from the experiment is an image. Using this image; we can visualize the results and intensity per spot. Each spot may give us the gene expression quantity through the measurement of the intensity of the data per probe. The images get segmented after identifying which pixels belong to a spot, statistics may be applied to summarize values such as averages, medians and intensity on the pixels in the spot [49].

The data we have gathered from this process can now be turned into a gene expression matrix. This matrix holds the values given for each spot on the microarray. The two most common ways of studying gene expression matrices is to either compare the expression profiles of the genes based on the columns or on the rows in the matrix. Using Euclidean or Correlation-based measuring we can find the similarity between the genes in the matrix. When handling massive amounts of expression data it can be difficult to find patterns which may yield any beneficial results. One of the common ways to make more sense of the data is to create subgroups based on the similarity it displays. This process is called clustering and has many different algorithms for grouping similar data together. Some of the issues facing such tasks are determining how many groups should be created and which clustering algorithm provides the best result. One of the large benefits of classification algorithms is the ability to identify clusters corresponding information of medical significance i.e. of clusters of cancer types or subtypes. The ability to discover information about the human body based on tissue/cell samples have become a reality. Much of this is due to clustering algorithms able to logically identify important patterns in gene expression data which can be linked to medical conditions or other biological functions [6, 32].

RNA Sequencing

RNA sequencing has a similar function to microarray where one can analyze cDNA to determine how which genes are expressed and at what level in a sample. Where microarray however is bound to a known library of genes, RNA sequencing is not restricted to previously detected expression of already known genes. In RNA sequencing the cDNA will be read into the system allowing for many new possibilities for how to handle the data. A known gene database will allow mapping of cDNA sequences a known reference genome. This will give similar results return to the traditional microarray, although often with more accurate estimates. Some of the larger benefits of RNA sequencing is its ability to run de novo analysis of new species for which

there are no reference genome. Novel genes may also be discovered in species where the reference genome is already known. This allows for much more versatility in uses where one is not limited to only analyze cDNA that already has been mapped. Genome analysis can also be rerun at a later time with updated genome mapping, providing the possibility of new important discoveries at virtually no additional cost [7-8,33].

Gene Expression in samples consist of different cell/tissue types

Gene expression varies from cell-type to cell-type (and from tissue to tissue). Some genes are more or less specific for a cell-type, or showing lower or higher expression in other cell-types or tissues. Other genes may be expressed in all cell-types and in all tissues. There are also variations in gene expression in the same cell-types/tissues.

When we see difference in expression between two samples it may indicate that the samples have different composition of cell/tissue-types and/or that the gene expression in one or more of the cell/tissue-types have changed [44,45].

Deconvolution

Deconvolution is a mathematical way to reverse the effect of mixed data. In gene expression this refers to handling global expression data that often come from mixed tissues or genes data that need to be separated in order to be useful for analytical purposes. To process these global expression data biologically, it is often very important to disentangle the cell types due to their different gene expression profiles. This allows the data to be analyzed on a cell-type level. Properly separated gene expression values may give important insight into the biological

environment in which genes are present and also in what proportions. This allows us to analyze what effect these genes may have in the sample area.

Two different approaches to separate the data are to either provide a marker gene list or to cluster the data on the subgroups which are found to be similar in the data.

A marker gene list is a collection of cell-specific signals. These signals can be used to estimate cell-specific signatures, cell proportions or cell-type related differential expression. Many of the marker genes have been collected through previous studies and are mostly well known and defined, yet there are still many cell type markers missing. Using this information will give a sort of roadmap for the deconvolver which indicates which genes belong where. Categorizing gene expression based on its similarity to these marker genes gives the clustering algorithm reason for comparisons. It will not, however, discover new cell groupings which are not a part of the marker gene list. We call this partial deconvolution. With only one unknown factor, this becomes a regression problem. Which is solved by the deconvolution algorithm by calculating the genes based on these estimated cell proportions from the marker genes [46].

Unsupervised clustering does not use a marker gene list and relies on the similarity within the data, as mentioned in the microarray section, in order to categorize the expression data. We call this complete deconvolution. It does not use known gene expression profiles but calculates these values based on the global expression data, to indicate which genes should be categorized where. The process relies on the assumption that the genetic expression will follow a certain pattern which we can identify and categorize from. The benefit of unsupervised deconvolution is its ability to find new gene expression profiles. The flexibility allows for gene

discovery and has a potential to return more valuable information assuming it can achieve a high enough accuracy in its clustering [11,12].

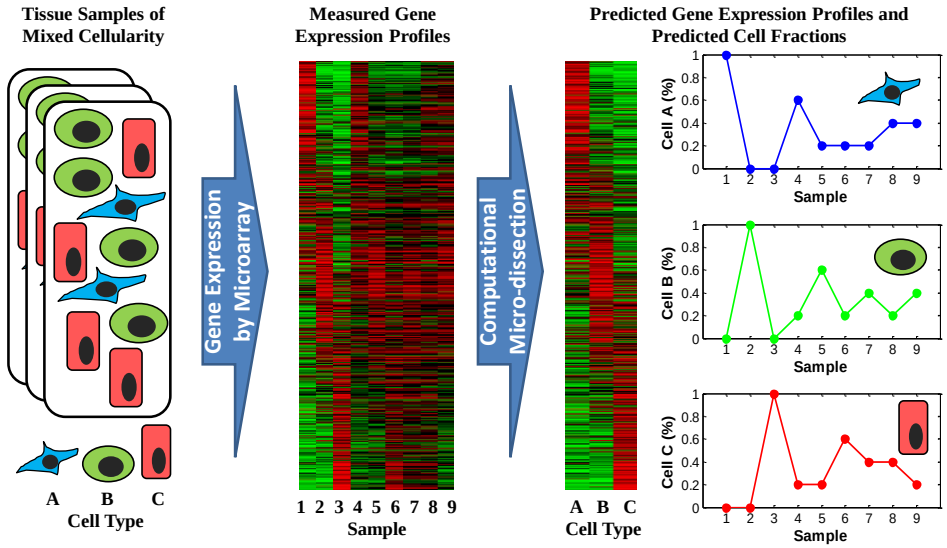

Figure 2. Visual example for each stage of the deconvolution process from a mixed group of cells through the separation process. The cell type expression pattern can be seen to the right of the sample, and we can see that sample 1,2 and 3 are pure as they fit perfectly with the estimated expression profiles, giving 100% correlation. This example has very "pure" data, in real biological data we do not expect 100% correlation [10].

In a linear model deconvolution, the mixing process can be shown with this matrix notation:

$X = SA$

$S = n \times k$

The matrix in S contains the genes in the sample (n), and the cell/tissue types (k).

$A = k \times p$

A contains the matrix k X p, same as in S, k contains the cell/tissue types and p contains the samples.

$X = n X k = (n X p) X (k X p)$

X then contains the measured gene expression levels of the mixed samples.

As discussed previously, if this is a partial deconvolution then either S or A will be known whereas if it is a complete deconvolution then neither will be known. Complete deconvolution is often referred to as unsupervised deconvolution.

In supervised deconvolution, A will be calculated using a marker gene list which is provided. This list is comprised of genes which are highly expressed in specific cell/tissue types and less expressed in others allowing the program to focus on data the user specified as more valuable for the deconvolution. Sometimes the marker gene list is not available or the user does not want to be limited to a prewritten marker gene list which does not incorporate all the data in the sample, in this case unsupervised deconvolution is used. This approach uses clustering algorithms on the gene expression dataset and uses this to create profiles which represents the different genes in the dataset. These profiles functions as the marker genes in the semi-supervised deconvolution instance. The benefits are that one is able to create an expression profile on the data in the dataset. This may be beneficial in finding new marker genes and to not be limited to previous gene discoveries. Negatively, it can cause weaker gene profiles which does not fit the proper pattern compared to focusing on the tried and tested marker genes [13].

Gene Expression and Health

Gene expression profiling through deconvolution is vital when investigating the genetic link to complex disorders and tumors. It has long been known that an individual's genetic profile is a

key contributor to many complex diseases. Up until recently there has been little opportunity to investigate the genetic links due to technical limitations. With the emerging technology, we can now break down the active roles that underlying genes are playing in these disorders. We are at a rapid pace finding new disease-causing mutations due to changes in the epigenome. By mapping and analyzing this information, we can both predict the likelihood of disease and potentially cure diseases by interfering with these processes [34].

Genetic traits that are caused by heritable genetic predisposition to different diseases have also been discovered. Using this information we have an opportunity to look for these traits and potentially detect such diseases early, giving a better chance of successfully treating the disease. We have long known that hereditary abnormal genes passed from parent to child gives a higher chance of getting different diseases later in life, but it's often been difficult to know which genes are causing it. By analyzing tumor tissue or other tissue associated with the disease we have the opportunity to find patterns of which genes expression patterns are present and in which proportions at different stages of the disease. The genes which are most linked to breast cancer is BRCA1 and BRAC2 which are present in our body to repair cell damage and to keep cells growing normally. When these genes mutates, they are known to cause problems, they are estimated to be the cause of 1 in 10 of all breast cancer cases [35].

The mutations in these genes are known to be inheritable, although if an individual has this mutation it doesn't automatically mean she gets breast cancer, but it does raise the odds of you getting it. This means that a test for the expression of these abnormal genes can be a very good starting point in determining individual risk factors in regards to these types of breast cancers. As our technology is improving and we are able to analyze larger and more varied amounts of genes from a more diverse population groups, we are finding complex correlations between multiple genes and different stages of a disease. There are discoveries indicating that certain cell variations are responsible for the predisposition of a disease while other cell variations are

responsible for the progression of the disease. Seeing the complexity of these issues, being able to sample, deconvolve and analyze massive amounts of genetic expression data effectively and accurately is paramount. We are facing a new medical frontier and understanding our genes are the cornerstone of this new endeavor [36].

Precision Medicine and Deconvolution

As our knowledge increases about genetics and medicine, and we have found that utilizing gene expression to understand biological functions for medical purposes is bearing fruit. We have for instance found that tumor heterogeneity is increasingly becoming a challenge in the field of cancer therapy. Both inter-tumor heterogeneity, the genetic difference in tumors between patients, and intra-tumor heterogeneity, the genetic difference within a single tumor. Tumor microenvironments are proven to be highly complex, with both cancer cells and non-malignant cells inhabiting the tumor space. These gene expression differences have been proven to affect how tumors react to different cancer treatments, which explains how individual patients with the same cancer diagnosis may have widely different outcome with the same treatment. As part of a new line of targeted disease prevention, a new treatment style is emerging called precision medicine. The use of gene expression profiles in medicine is not limited to cancer tumors, but has found many important medical uses. It is all a new approach to medicine which assumes individual patients may have varied needs based on which biomarkers are present. Biomarkers are according to the World Health Organization (WHO) “Any substance, structure or process that can be measured in the body or its products and influence or predict the incident of outcome or disease” [37].

In order to make precision medicine viable there is a need to increase the library of known genetic biomarkers and how their presence affect the individual. A way to do this is to map gene expression profiles in cancer tumors in order to find medically valuable patterns in the patient

groups. To accurately map these biomarkers there is a need for large databases with different tumor variations allow one to a molecular fingerprint, indicating the unique reproducible genetic properties. Utilizing this information one can design drugs specifically for the individual patient which are designed to work optimally when targeting tumors with certain gene expression profiles. In order to build up this library of biomarkers we need to get the gene expression profile for these tumors in an effective manner. One of the big issues currently, is the lack of generalizable biomarkers which can be applied to most patients. Utilizing big data the hope is to effectively find patterns which may allow effective categorization of disease. When generalized disease profiles are achieved, one can subcategorize it into different variations allowing for further information indicating expression differences and its effects on the general disease [38].

BACKGROUND

Deblender/Digital Sorting Algorithm (DSA)

Deblender, the deconvolution tool my program is created to be an add-on to, but as it is not yet published, I have to limit discussion of the product in my thesis and will focus on the parts I do use. The parts of Deblender I use for my program are mostly identical to the published DSA tool, I will reference Deblender but everything stated is also true for DSA.

Deblender is a complete deconvolution tool with the ability to both analyze gene expression data in a semi-supervised or an unsupervised form, clustering is only used in the unsupervised process. Deblender has the ability to analyze microarray data and RNA sequencing data, using K-means or K-mediods to cluster the data. One of the base assumptions made in Deblender is that a subset of genes is highly expressed in specific cell/tissue types and lowly in others. Using this assumption, the tool can estimate the mixture proportions of the cell/tissue samples.

Deblender will assume that S_m, a $m \times k$ matrix has a set of highly expressed marker genes or clusters in their respective cell/tissue types and are not or lowly expressed in the remaining cell/tissue types.

$$X_m = S_m A$$

$$S_m = \begin{pmatrix} g_{11} & 0 & \dots & 0 \\ g_{21} & 0 & \dots & 0 \\ 0 & g_{32} & \dots & 0 \\ 0 & g_{42} & \dots & 0 \\ 0 & g_{52} & \dots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \dots & g_{mk} \end{pmatrix}$$

$$\tilde{S}_m = \begin{pmatrix} m_1 & 0 & \dots & 0 \\ 0 & m_2 & \dots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \dots & m_k \end{pmatrix}$$

In the S_m matrix we can see that each cell/tissue type has their sets of highly expressed marker genes/clusters cluster which are not expressed in the other cell/tissue types. In our Deblender/DSA application, instead of averaging the marker genes expression profiles and use these average profile, they are substitutes by cluster representatives.

Deblender calculates $\tilde{S}_m$, from S_m by taking the average of the mixed gene expression profiles for each of the marker genes/clusters. Using this we then get this equation:

$$\tilde{X}_m = \tilde{S}_m A$$

The $\tilde{S}_m$ matrix is a diagonal matrix, this allows us to multiply both sides with $\tilde{S}_m^{-1}$, which gives this equation:

$$\tilde{S}_m^{-1} \tilde{X}_m = A$$

In our model we have a constraint that says that the sum of proportions for each sample sums to 1 and each proportion is larger or equal to 0, which allows us to create a new equation with k unknown variables (diagonal elements of $\tilde{S}_m^{-1}$) [13,41].

$$\sum_{i=1}^{k} \left(\tilde{S}_m^{-1} X_m \right)_{ij} = 1$$

Deblender Unsupervised

Deblender's unsupervised mode as it has been implemented in my tests, requires these three steps: Preprocessing the data, clustering and identification of the tissue/cell type.

Preprocessing the data: The data need to be raw-normalized values without log transformations. Deblender recommended for filtering un-annotated probes. If there are multiple probes per gene identifier, choose the one with the highest variance. Users can also apply a percentage cutoff in order to filter genes with high or low expression vector norms.

Clustering: The two clustering algorithms available in Deblender, K-means and K-mediods, using correlation as the distance function, both linear and log scale clustering, are available. Both of these clustering algorithms are well suited for the large datasets they are used for.

Identify the cell/tissue type: After the deconvolution process is finished, the last step is to connect the clusters to the cell/tissue type. For the unsupervised mode there is no additional information available to indicate to which cell/tissue type each cluster should be connected to. Checking all possible combinations recording the correlation compared to known proportions of the different cell/tissue types. The combinations getting the highest correlation becomes the retained values [13].

Clusters

Clustering is grouping data based on observed patterns, using one or more clustering algorithms. Gene expression profiles are determined based on the genes expressed to give a picture of the cellular function. For our purpose, we approach the data with the expectation that we have an unknown number of profiles, which our data will fit into. The assumption is that each individual gene will have a unique gene expression profile which is consistent enough so that the data can be grouped according to which gene profile it belongs to. Not knowing the number of expected clusters creates a few potential problems as the data may not pick up each expression profile as unique. If the expression profiles are too similar it could merge two profiles or split a cluster into multiple profiles because of the cluster not finding the profile consistent enough. In DeCloud the tool attempts to find the gene expressions patterns in the data and to use these patterns to separate the data into the gene expression profiles which makes up the individual clusters. For this purpose we do not preset the specific number of clusters we are expecting but rather set a range of possible clusters. This allows the clustering algorithm to separate the data into the number of clusters it estimates to be most accurate based on similarity. We then calculate the relevance of these clusters using the RankAggreg package to estimate which clusters are created by the best profiles. RankAggreg does this by ranking the clustering algorithms and clusters created in our range of clusters by three different clustering

criteria, Connectivity, Dunn and Silhouette width. By using this unsupervised approach we are attempting to discover patterns in the data which are not limited to a known list of expression profiles and which may be more accurate as excess clusters not meeting the threshold may be discarded [14-16].

For our clustering, we are using the two different clustering algorithms to handle the microarray data in our results, Agglomerative Hierarchical Clustering(Hierarchical) and Partitioning Around Medoid Clustering(PAM). When choosing which clustering algorithm to use its important to know which type of data is being used. The microarray data and RNA sequencing data are dissimilar due to RNA sequencing being count-based while microarray is not. This potentially reduces the accuracy of the microarray clustering algorithms, while the RNA sequencing algorithms for clustering are only designed to work with whole number and as such will not work with microarray data unless they have been transformed into whole numbers. The RNA sequencing count-based data is created by recording the number of sequence fragments assigned to each gene per sample. They are reported to produce low noise data, which may help detecting transcripts when there are low expression levels. We have the ability to calculate RNA count data in the optCluster package, but due to lack of proper count data in the test sets available, they are not presented [47].

In the optCluster package we have access to ten different clustering algorithms intended for handling microarray data. The list is as follows: “Hierarchical”, “Agnes”, “Diana”, “K-means”, “Pam”, “Clara”, “Fanny”, “Model”, “Som” and “Sota”.

There are also six different clustering algorithm variations intended to cluster count data from RNA sequences. These clustering algorithms are: “Non-Binomial EM (Expectation Maximization)”, “Non-Binomial DA (Deterministic Annealing)”, “Non-Binomial SA (Simulated Annealing)”, “Poisson EM”, “Poisson DA” and “Poisson SA” [18].

METHODS

DeCloud has many clustering algorithms, which offers diversity and increases the usability and potential for the application. The results however narrows the focus down to only two of the clustering algorithms, Hierarchical clustering and PAM clustering. Hierarchical clustering was a natural choice as, it gives a different approach to clustering compared to K-means and K-mediods clustering which is implemented in Deblender. PAM was the second clustering algorithm chosen as it had an already implemented silhouette width calculation in the application, which is used to create a novel approach to handling clustering data, referred to as a data filter function. The PAM clustering algorithm is similar to K-means and K-mediods which allows for a more direct comparison when a data filtering method is implemented which I have created. More on the data filtering implementation in the “Filtering data through clustering” paragraph.

Agglomerative Hierarchical Clustering

Hierarchical clustering can be illustrated by dendrograms which will give a visual view of the clustering structure. The clustering algorithm starts with each of the observations in the dataset as its own individual cluster and merge them together until all the data are merged into one big cluster. This means that for each step the two clusters are merged. In order to extract the number of clusters one needs the cluster is “cut” at the correct height in the dedrogram where the cluster width is the given number needed. There are four options available to decide how to handle the distance used. The four methods are: Average linkage, complete linkage method, single linkage method and Ward’s method.

Average method: Uses the average pairwise distance between cluster objects to determine how close the clusters are together. The two clusters with the smallest average distance between objects per cluster is combined per iteration.

Complete linkage method: This method looks at the maximum distance for each cluster combination, it then mergers the clusters with the lowest maximum pairwise distance together for each iteration.

Single linkage method: This method finds the two clusters with the lowest minimum pairwise distance and combines them for each iteration.

Ward's method: Calculates the within-cluster variance per cluster combination and merges the two clusters to give the smallest increase in the total within-cluster variance per iteration [18].

PAM (Partitioning Around Medoids) clustering algorithm

PAM is one of the medoid-based clustering algorithms. For mediod-based clustering algorithms data points are used as centers for clusters, compared to centroid-based clustering algorithms not bound to a data point but a cluster location. To initiate this clustering algorithm, it is necessary to set a given number of clusters and assign a data point to be a mediod, also known as a cluster center, to be the center for each assigned cluster.

Once the initial medoids have been set, the clustering algorithm will assign all the data points to its closest medoid. Once all the data points are set to a mediod, a new mediod will be assigned for each cluster, and the closest data points will be assigned to this new mediod.

The goal is to reduce the dissimilarities of the data points to their given mediod. When the clustering algorithm can no longer find a new mediod which can reduce the sum dissimilarity, the clustering algorithm ends and the lowest dissimilarity clusters are selected [17,18].

K-Means clustering algorithm

K-Means is another iterative clustering algorithm which starts with a given set of clusters and cluster centers. K-means have cluster centroids, a centroid is a cluster center, which does not need to be a data point, but is tied to a location. The data points will then be assigned to their closest centroid. The average of the observations for each centroid computes new centroid locations. The process of averaging observations to each centroid continues iteratively until the maximum number of iterations is reached and the iteration with the lowest sum distance will be chosen [18,19].

Cluster Assessment

To ensure good clustering results it is important to be able to evaluate the clusters that have been created. A good clustering algorithm should allow a function to evaluate how statistically relevant the clusters are based internal or external data.

In our program we use the intrinsic validation measures which give us an insight into the statistical properties of the clusters. The three validation measures used are connectivity, silhouette width and Dunn Index. Connectivity is an intrinsic validation measure that analyses how neighboring data points are grouped or connected. The neighboring data points are calculated based on the distance function provided; in our case, correlation distance is used [18].

Calculating the connectivity of the clusters it is done by determining the *n* closest data points per data point. 1/n is added to the total connectivity value if the nth closest data point belongs to a different cluster than the first data point. The value range is between zero and infinity, where the lower the score is the better the result [28].

Silhouette Width differs from the connectivity validation measure in that it looks at cluster separation and the compactness of the clusters [18].

Silhouette values are calculated by this formula:

$$silhouetteVal = \frac{b_i - a_i}{\max(a_i, b_i)},$$

a_i is the average distance between a data point and all the other data points in the cluster.

B_i is the average distance between a data point and all the data points in the closest neighboring cluster. This calculation as a value range between -1 and +1, where higher values are better results [28].

Dunn Index is similar to the Silhouette Width validation measure in that it also analyses cluster separation and cluster compactness.

The Dunn Index takes the minimum cluster distance between two clusters and divides it by the maximum cluster within the same cluster. In the event of more than two clusters, the two clusters with the smallest distance between each other will be picked and will be divided by the cluster with the largest diameter. The diameter for a cluster is determined by the distance function used. In our calculations, it is used with correlation distance for cluster evaluation and with euclidean distance for tbe data filtering function. The value range is from zero to infinity, higher scores indicates better results, although in my filtering function Dunn Index is implemented such that the lowest Dunn Index score was the best result. This is an unorthodox implementation of the Dunn Index and does not follow the general application of the method [18,28].

Calculating best cluster set and clustering algorithm

RankAggreg is a package built into optCluster and offers an effective method for utilizing multiple assessment measures in unison. When validating clusters we are often stuck trusting one validation measure instead of a combined performance estimate of all the given measures. RankAggreg is able to not only provide an average score of all measures for each clustering algorithm used but also for all numeration of clusters. This allows us to analyze a range of clusters with different clustering algorithms and to find which number of clusters combined with the clustering algorithms performed the best total. In DeCloud the RankAggreg portion of optCluster calculates which cluster set is the best after all the cluster sets are created. It also calculates which clustering algorithm performs the best on the sets as well. A cluster set, for the purposes of this paper, is the groups of clusters created each combination. When optCluster has the input to calculate 2:8 clusters, each set incrementing by one cluster starting at 2 clusters, last cluster set holds 8 clusters. Due to the difficulty of data sets chosen for testing only data set A relies fully on the RankAggreg implementation to calculate optimal cluster set, but this is due to the complexity of the data sets chosen in data set B and C.

RankAggreg can use either the cross-entropy Monte Carlo algorithm or the Genetic algorithm. It also gives the option between two different distance functions, weighted Spearman's footrule or the weighted Kendall's tau distance in order to stabilize the aggregation algorithms. Both distance functions being modifications of the distance functions Spearman and Kendall [18].

The weighted Spearman's footrule takes the absolute differences between the ranks of all the unique elements from two ordered lists. The smaller the value of the difference, the more similar the lists are. The weighted portion of the calculation is a penalty system which takes the sum of movements within the lists being calculated.

The weighted Kendall Tau's distance also measures the distance between two ordered lists. If the ordering of two values, t and u, are the same in both lists, then no penalty is given. If t is ahead of u in one list, but opposite in the other, a penalty of 1 is incurred. The weighted part of the function is set as the absolute difference in the scores of t and u. The lists used are normalized between 0 and 1 [48].

In RankAggreg Cross-Entropy and Genetic Algorithm are the algorithms available to go through all the scores for each cluster and cluster set per algorithm. Using this information, it ranks each combination of cluster algorithm and cluster set from best to worst.

Genetic Algorithms (GA) was used for all the tests referred to in the results section, yet extensive testing was done using cross-entropy (CE) and no substantial difference was found in the final product. My choice of using Genetic Algorithm was simply due to the runtime comparisons as Cross-Entropy showed itself to be slightly slower for these particular tests.

Cross-Entropy Monte Carlo Algorithm

Originally developed in order to calculate the probability of rare events, it later was proposed as a valid method for calculating weighted rank aggregation based on lists made up of different cluster validation measures.

The Cross-Entropy Monte Carlo Algorithm creates an $n \times k$ matrix where all entries are 0 or 1. Total number of unique clustering algorithms is n and the constraint of the matrix is that both column sum and row sum must equal at most 1. Due to this all the variables follows a multinomial distribution, and reading from left to right the position of the value 1 in each column determines the order of the ranked list.

The algorithm is broken into four steps:

1. Initialization: This step is setup with an initial matrix of parameters making sure all the clustering algorithms have an equal chance at being picked in each *k* position of the ranked list.

2. Sampling: Here a random sample is selected from the most recent parameter matrix. Using this an optimal list and objective functions values are calculated.

3. Update: Using the last sample and objective functions values, the parameter matrix is updated. A new sample is then created from the new matrix and the process begins again with the goal to minimize the objective function values.

4. Convergence: The update function continues until the optimal list stays unchanged for a set number of iterations This optimal list is then returned as the chosen optimal list of scores to use [18].

Genetic Algorithm

Genetic algorithm uses an evolutionary concept to achieve the optimal solution, Charles Darwin's survival of the fittest evolutionary theory. The algorithms assume that the "fittest" solution will "survive" through all the iterations, effectively making the chosen solution the evolutionary winner. It may even develop the "fittest" solution during the run.

The algorithm has five steps:

1. Initialization: First a predetermined population size must be chosen. Multiple randomly generated, equal length, ordered lists of possible solutions are then created based on the population size input. The larger the population size, the higher the chance of finding the optimal solution.

2. Selection: A fitness function is used to evaluate each individual lists of possible solutions. Based on the scores received by the fitness function, a weighted random sample is used to create a new group of solutions, removing the lower scoring list.

3. Crossover: A crossover probability is set and each list in the new group of solutions will be subjected to a one-point crossover. A point is calculated based on the crossover probability, then all the elements of each list ranked lower than the specified point will be swapped with another list, this is done to all the lists with elements under a specified point.

4. Mutation: A mutation probability is set, this probability will determine how often an element will be randomly altered in the lists. One or more elements from any of the lists in the group of solutions may be altered based on how high the probability is set.

5. Convergence: In the convergence step, the algorithm will iterate through the selection, crossover and mutation until a list remains the optimal for a set number of iterations. This optimal list will then be presented as the chosen list for the algorithm [18].

The optCluster package

The optCluster package is a tool which covers a lot of ground when handling clustering data. Most R packages used to calculate clusters are mostly limited to few clustering algorithms and are designed for certain datatypes. The optCluster package is built around the idea that there are many packages available with the ability to fulfil the tasks asked of them, such as calculating cluster, validation measures or determining the optimal cluster number and clustering algorithm.

optCluster is a combination of packages, creating one streamlined package which can handle variations of data and needs. The three main packages used are clValid, RankAggreg and

nbclust. The original optCluster package was released with only clValid and RankAggreg used but later nbclust was added for its ability to handle RNA data [18,29].

Implementation

DeCloud as a tool is implemented in R, taking advantage of the libraries available in order to handle complex data. Running clustering algorithms on big datasets is very resource demanding and I had to move DeCloud over to the university server cluster named "Kjempetuja". Even with the help of such a powerful platform, I found that certain clustering algorithms available in optCluster were not optimized to a great enough extent to functionally cluster the large datasets. Therefore I had to choose the clustering algorithms most useful for my purposes.

When starting my project I was informed of the intent for Deblender to be ported from Matlab to R, which is why DeCloud was written in R instead of Matlab. Due to unforeseen circumstances, the team building Deblender was not able to port the application in time before my research was done. DeCloud is intended to become an add-on to Deblender as to hopefully improve the tool's flexibility and to increase its potential. This unfortunately forces me to run my program on multiple platforms. The Matlab datasets used in the results are from Deblender and are read into my code. The data then gets transformed into data useful to R, and then transformed back into Matlab datasets fully clustered by DeCloud. This should not be a necessary step when Deblender gets ported to R, assuming they choose to implement it.

The tool itself utilizes the optCluster as the basis for the calculations and will mostly lean on the built-in evaluation measures for internal validation, such as Connectivity (to a lesser degree for my analysis), Dunn index and Silhouette width. RankAgg will give a consensus estimate based

on the three different measurements and decide which clustering algorithm performs the best and estimate how many clusters are optimal. There is a built-in function indicating if the data analyzed are count data or not, which allows for the use of the variations of Non-binomial and Poisson algorithms. If the count data function is set to true, the program will not run if the data is not in count data format. All clustering algorithms, except for Deblender's K-means and K-mediods, run from 2 to 8 clusters, 2 clusters being the smallest possible set and 8 being an arbitrary number high enough for our purpose.

Filtering data through clustering

For very noisy data, RankAggreg may need to change how it weighs the internal validation measures for calculating the optimal clusters number.

The idea behind the implementation was to use clustering as noise filters to catch data that were less useful for our purpose in order to find better patterns in data left over.

Using the optimal cluster number (k) calculation, provided by the optCluster package, I looked at the sets of clusters created which had more clusters than what was estimated to be optimal (k). The system keeps the top (k) clusters within each cluster set according to silhouette width, discarding unused clusters. This assumes that high scoring silhouette-width clusters hold more valuable information than the other clusters in each set. This gives a top (k) set of clusters for each set of clusters larger than the calculated optimal cluster number. In order to determine which top (k) set of clusters contained the better clusters, a penalty system was implemented. For this it assumes that due to high silhouette-width scores, the clusters hold valuable information and that overlapping data based on Euclidean distance may be an indicator of good data. Another assumption was that large diameter clusters in Euclidean distance were beneficial as they held the unique parts of the clustering profiles where they do not overlap with other

clusters. To achieve this, a penalty method were implemented to encourage the maintenance of partial cluster overlap structure and maintain non-overlap data in the clusters while still allowing the removal of noisy data. This was done by reversing the approach to the Dunn Index, where very compact and separated clusters are preferred. The implementation favored the top (k) clusters which gave the lowest Dunn index, opposite of its traditional use which wants to maximize the Dunn Index score. Using this system it was possible to remove noisy data from the clusters and find which of the cluster sets contained the best cluster results.

Pearson Correlation Coefficient

To evaluate the quality of our results we use Pearson correlation coefficient to measure the strength of the linear association between our estimated cell-type proportions and the real proportions. The Pearson correlation coefficient calculation produces a value between -1 and 1 where 0 means no correlation. A negative value indicates negative correlation and a positive value is positive correlation, a score of 1 means perfect correlation [50].

The datasets

For our purposes of comparison the original implementation of Deblender, we use three of the same datasets which were used to test its efficiency compared to other established deconvolution tools.

GSE19830: Microarray dataset which includes samples of pure rat brain, liver and lung tissue including 11 other different proportions. There are 3 technical replicates with 42 total samples, 33 of which are mixed tissue samples. For my results the 33 unknown mixed tissue samples was used to calculate the clusters and the 9 known samples were used to check the accuracy of the clustering.

GSE11058: Microarray dataset containing pure samples of immune cell lines Rajo, THP-1, IM-9, Jurkan and 4 mixtures of the aforementioned cells in different proportions. There are 3 technical replicates making up a total of 24 samples, even split between cell type-specific and mixed samples. For my purposes the unknown 12 samples were used for clustering. The accuracy of the clustering was measured against the known samples.

GSE65135: Microarray dataset containing 14 disaggregated lymph node biopsies from patients with follicular lymphoma. The dataset was analyzed to have B cells, CD4 T cells and CD8 T cells by flow cytometry.

RESULTS

In the results we analyze the three aforementioned datasets GSE19830, GSE11058 and GSE65135. For simplicity and readability, I will rename these Dataset A (GSE19830), B (GSE11058) and C (GSE65135).

On each of these datasets four different analysis were run, Deblender, Hierarchical, PAM and PAM with filter. For each dataset Deblender will be displayed first, followed by Hierarchical, PAM and lastly PAM with filter. At the bottom of each dataset analysis, a summation comparing all four tests will be shown.

Although not published yet, Deblender is a good deconvolution tool which can perform at the same level or better than other deconvolution tools published. For the datasets represented here it is proven and those results are pending publishing. As the purpose of this thesis is to create an addition to Deblender, it seems appropriate to use it as a baseline to compare my results.

GSE19830 (A)

For the A data set we see good results in Deblender which uses K-means to calculate the results. The data set has three tissue types, liver, brain and lung. The mixed data set has 33 unknown tissue samples (columns) and 20638 rows of unknown gene data (rows), the data have been filtered in Deblender prior to clustering though a high/low expression filter. My package acquires the data from Deblender before the clustering step to ensure identical data for all tests.

Dataset A Deblender

Deblender utilizes K-means for its clustering with correlation distance. The results indicate strong correlation between the estimated clusters and the true values.

Tool/Algorithm	Liver Tissue	Brain Tissue	Lung Tissue
Deblender/K-means	r : 0.962	r: 0.983	0.983

Table 1. Represents Pearson correlation score for Deblender for each tissue type in Dataset A.

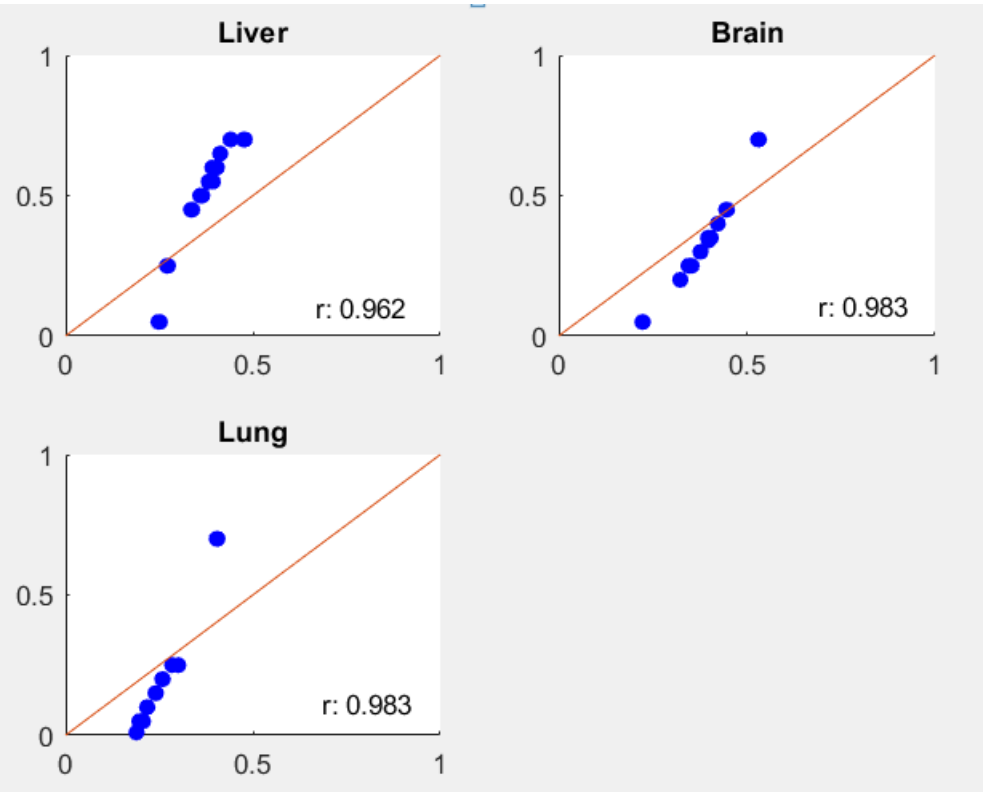

Figure 3. Scatterplot showing correlation for the estimates proportions compared to the real values in the A dataset.

Dataset A DeCloud Hierarchical

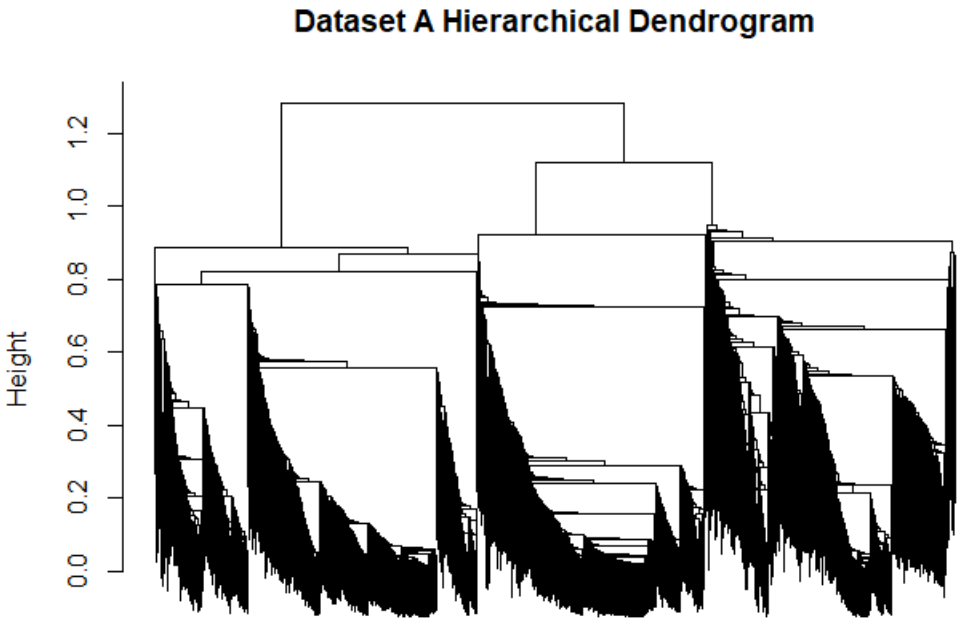

Figure 4. Hierarchical Clustering Dendrogram for A.

The Hierarchical test uses the same data as Deblender for the test. We can see in Figure 4 how the clusters combine from individual datapoints at the bottom in the figure to combining into two clusters at the top. If we look at the dedrogram we can also see three clusters at height 1.0, after this the clusters split apart rapidly. RankAggreg calculated that three clusters was optimal for this dataset, using the intrinsic validation functions connectivity, Dunn Index and Silhouette. As we cut the dedrogram to return 3 clusters we get the results shown in the table 2 below.

Tool/Algorithm	Liver Tissue	Brain Tissue	Lung Tissue
DeCloud/Hierarchical	r : 0.982	r: 0.980	0.987

Table 2. Represents Pearson correlation score for DeCloud Hierarchical clustering algorithm for each tissue type in Dataset A.

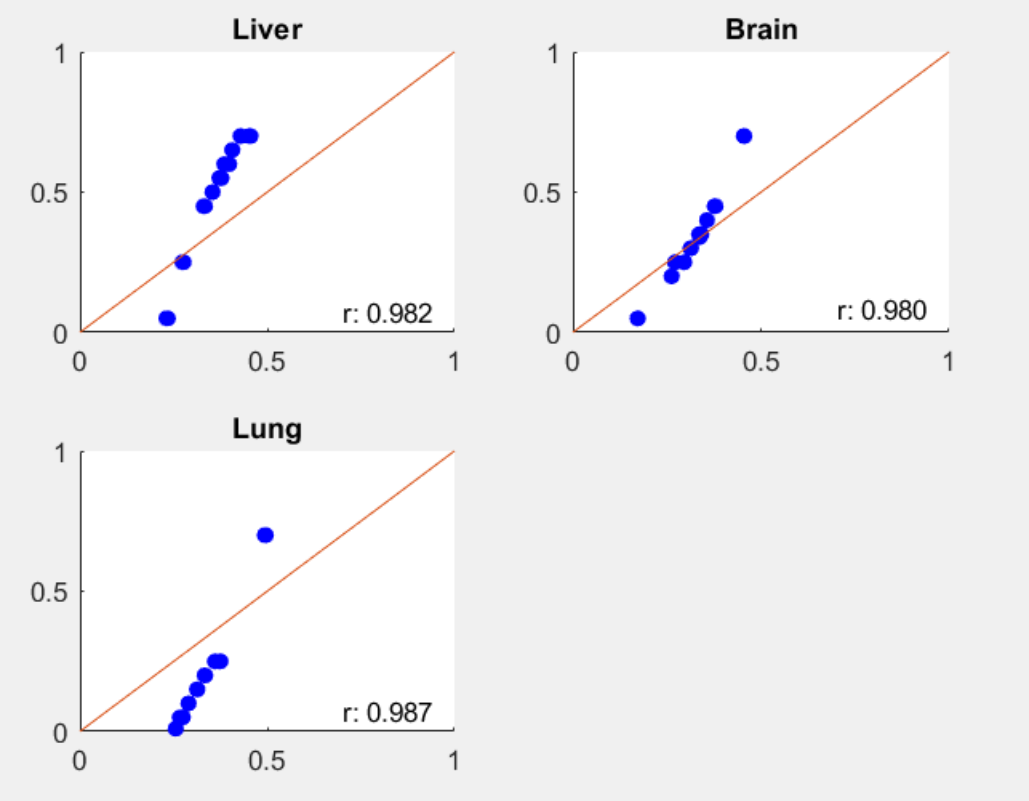

Figure 5. Above are three scatterplots representing the correlation of the Hierarchical clusters to the different tissue types.

The results indicates strong correlation in the tissue samples and are better than Deblender for 2/3 of the sample types. Lung tissue is slightly higher with r: 0.987 compared to Deblender's r: 0.983. There is a larger increase in liver tissue correlation from r: 0.962 to r: 0.982 and a slight decrease in brain tissue correlation from r: 0.983 to r: 0.980.

Dataset A DeCloud PAM

For the A dataset we see that cluster set with 3 clusters has the best Connectivity and Silhouette with and second best Dunn Index. The RankAggreg function in optCluster calculates that 3 clusters are the optimal number of clusters based on Connectivity, Dunn Index and Silhouette Width, same as hierarchical.

Tool/Algorithm	Liver Tissue	Brain Tissue	Lung Tissue
DeCloud/PAM	r : 0.956	r: 0.983	0.984

Table 3. Represents Pearson correlation score for DeCloud PAM clustering algorithm for each tissue type in Dataset A.

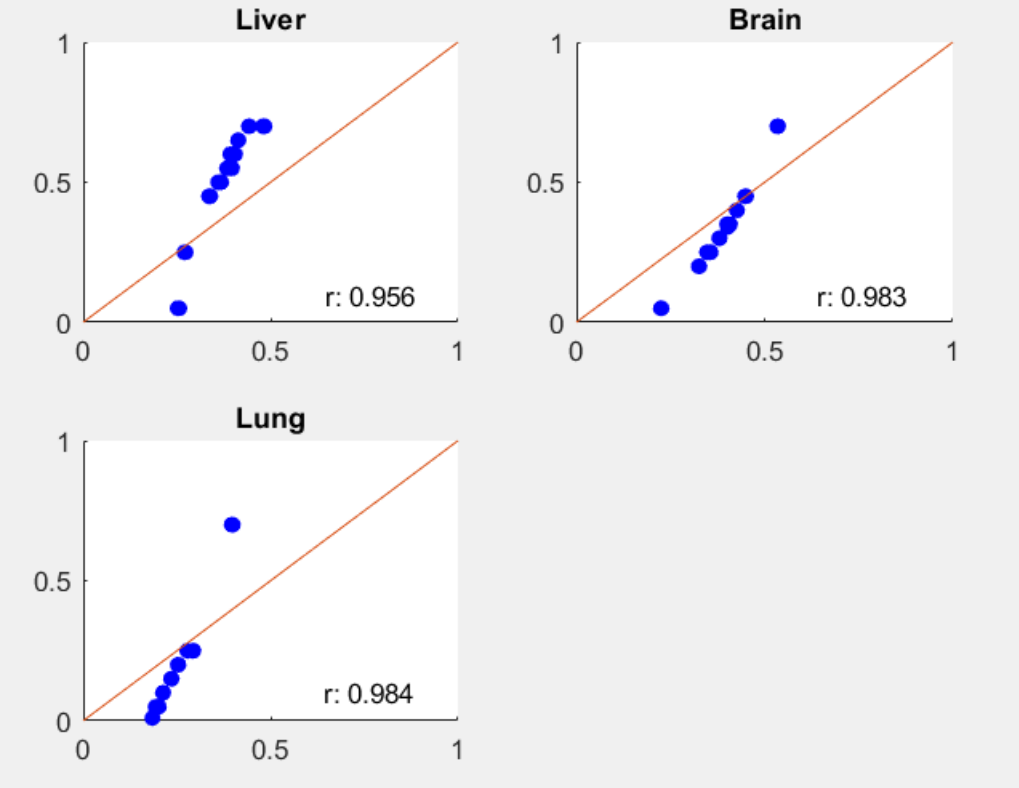

Figure 6. Above are three scatterplots representing the correlation of PAM clusters to the different tissue types.

The results are very reminiscent of the Deblender results, something, which makes sense since PAM is very similar to K-means. The differences are minimal, although the PAM clustering algorithm does underperform slightly compared to Deblender.

Dataset A DeCloud PAM with cluster filter

For the PAM clustering algorithm with filter, the 3 clusters with the highest silhouette width per cluster are kept, while the rest are discarded. The clusters sets with fewer clusters than what was calculated to be optimal do not get considered.

Cluster Set	1	2	3	4	5	6	7	8
3	**0.406**	**0.575**	**0.591**					
4	0.248	**0.441**	**0.585**	**0.280**				
5	**0.330**	**0.487**	**0.557**	0.259	0.084			
6	**0.343**	0.154	**0.431**	0.207	**0.498**	0.044		
7	0.164	**0.464**	**0.417**	**0.527**	0.199	0.025	0.005	
8	0.214	0.084	**0.410**	**0.524**	0.171	**0.294**	0.008	0.006

Table 4. Above is a table showing the Silhouette-width per cluster in cluster sets, 3-8. The 3 clusters with the highest silhouette width per cluster set are kept (in bold).

Using the average Silhouette-width per cluster, the three highest scoring clusters per set based on my observations in the data and testing are retained. It was found that this yielded the best result from each cluster set consistently, but did not necessarily indicate which cluster set was superior.

To find the best cluster set, I use the Dunn Index score in Euclidean distance and pick the cluster set with the lowest value. The Dunn Index is traditionally used as a measure to find highly compact and separated data where score should be maximized. Due to the data used and the implemented filtering system of choosing best silhouette width clusters, I found that overlapping data in clusters with larger cluster radius yielded better results. In this case, the lowest Dunn index is cluster set 4 with a Dunn Index score of 0.0123. This is slightly lower than cluster set 3 with a Dunn Index score of 0.0127. Based on observed clustering behavior, the Dunn index serves to penalize the cluster set when the data get too filtered-out. It has also been observed that when important data are getting removed, the data structure is altered and the range of data in the clusters are lowered. This is represented in the implosion of the cluster radius.

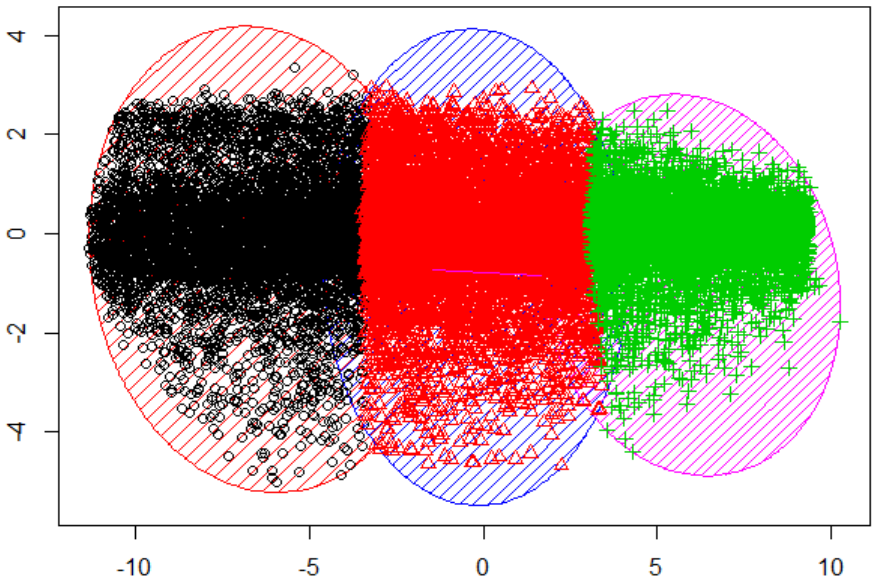

Figure 7. Cluster plot showing the clusters generated by PAM cluster set 3.

In figure 7 we can see the three clusters originally implemented by PAM on the A dataset. Here we can observe a slight overlap of data on both sides of the blue cluster (red data points) with its neighbors.

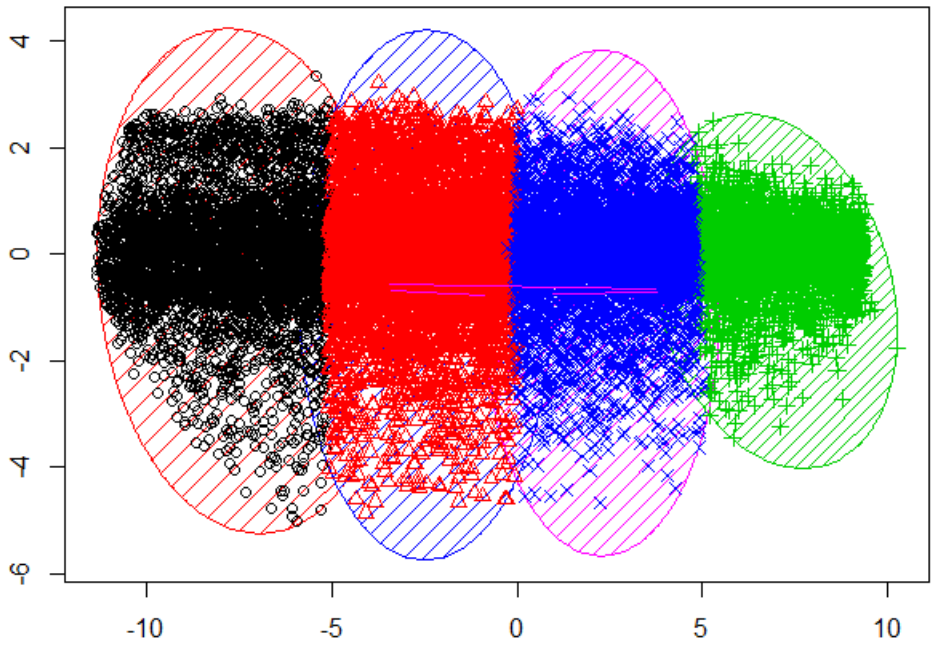

Figure 8. PAM 4 clusters, based on recommended optimal amount of clusters(calculated in the previous PAM test by RankAggreg) we know we have one too many clusters present in this cluster plot.

In figure 8 the cluster plot above we have the 4 PAM cluster set, the blue cluster with red data points has the lowest silhouette width indicating that it is the lowest confidence cluster. A low silhouette score indicates a higher likelihood that it holds data points that should have been in another of the clusters.

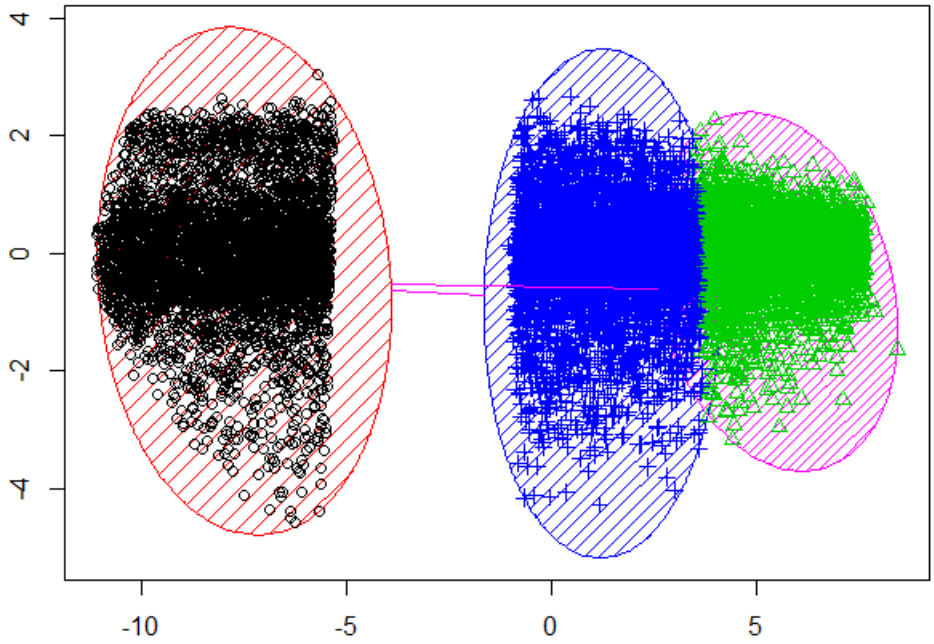

Figure 9. Cluster plot showing the GSE19830 clusters after the lowest silhouette cluster has been removed from the 4 PAM cluster set.

In figure 9 we can observe the clusters after the filtering process has taken place and the lowest silhouette width cluster has been removed, leaving us with three clusters to represent the estimated optimal 3 clusters.

Tool/Algorithm	Liver Tissue	Brain Tissue	Lung Tissue
DeCloud/PAM(w/cluster filter)	r : 0.989	r: 0.983	0.994

Table 5. Represents Pearson correlation score for DeCloud PAM clustering algorithm, with filter through clustering, for each tissue type in Dataset A.

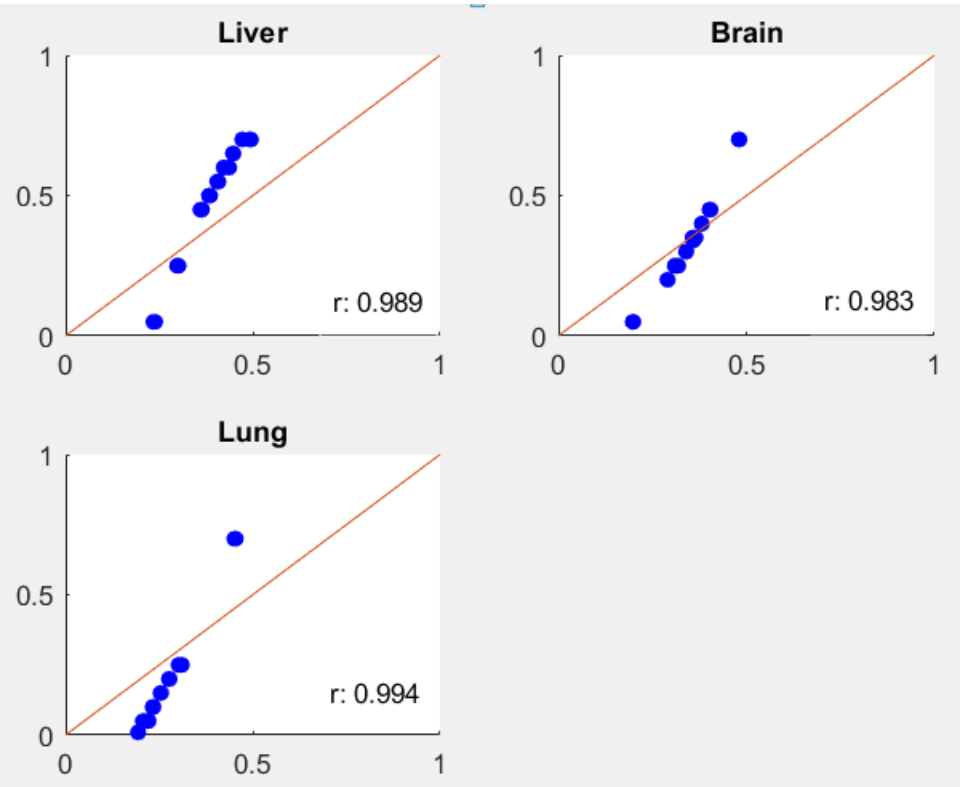

Figure 10. *Above are three scatterplots representing the correlation of PAM filtered clusters to the different tissue types.*

In Figure 10 we can see improvement in the deconvolution accuracy after implementing the filtering system. Liver tissue increases from r: 0.962 to r: 0.989, Brain tissue stays the same, while Lung tissue estimates increase from r: 0.983 to r: 0.994. Based on these results, PAM with cluster filter shows improvement even compared to the very accurate Hierarchical clustering algorithm.

Tool/Algorithm	Pearson total	RMSE
Deblender/K-means	r: 0.895	0.1459
DeCloud/Hierarchical	r: 0.897	0.1624
DeCloud/PAM	r: 0.888	0.1450
DeCloud/PAM (w/cluster filter)	r: 0.986	0.1352

Table 6. Total Pearson correlation for each result on dataset A, with RMSE.

In our total Pearson correlation coefficient calculations we see that Hierarchical and PAM with cluster filter are better than Deblender, PAM clustering algorithm which uses the filter implementation shows the greatest improvement over the other clustering algorithms.

GSE11058 (B)

The GSE11058 data set in its mixed data form with 41948 unknown gene data (rows) and 12 unknown tissue samples (columns). It is a very large dataset with a large amount of noise and overlapping clusters. This dataset has 4 cells, Jurkat, IM-9, Raji and THP-1. Due to the amount of noise in this dataset I am not able to reliably calculate the recommended number of clusters and have for the purpose of the analysis hardcoded in the recommended number of clusters. It is possible to get the correct number of clusters by changing how RankAggreg weights Connectivity, Dunn Index and Silhouette Width, but doing so without in-depth knowledge of the dataset would be very unreliable. Deblender also requires that the number of clusters is hardcoded in for this test due to the amount of noise in the data.

Dataset B Deblender

The Deblender uses the K-means clustering algorithm to cluster the data with mixed results. I have observed that dataset B, in its fairly unfiltered form, is very difficult for the traditional clustering algorithms to classify correctly. This is most likely due to very poor separation between the clusters and overlapping data.

Tool/Algorithm	Jurkat	IM-9	Raji	THP-1
Deblender/K-means	r : - 0.458	r: 0.866	r: 0.497	r: 0.985

Table 7. Represents Pearson correlation score for Deblender for each tissue type in Dataset B.

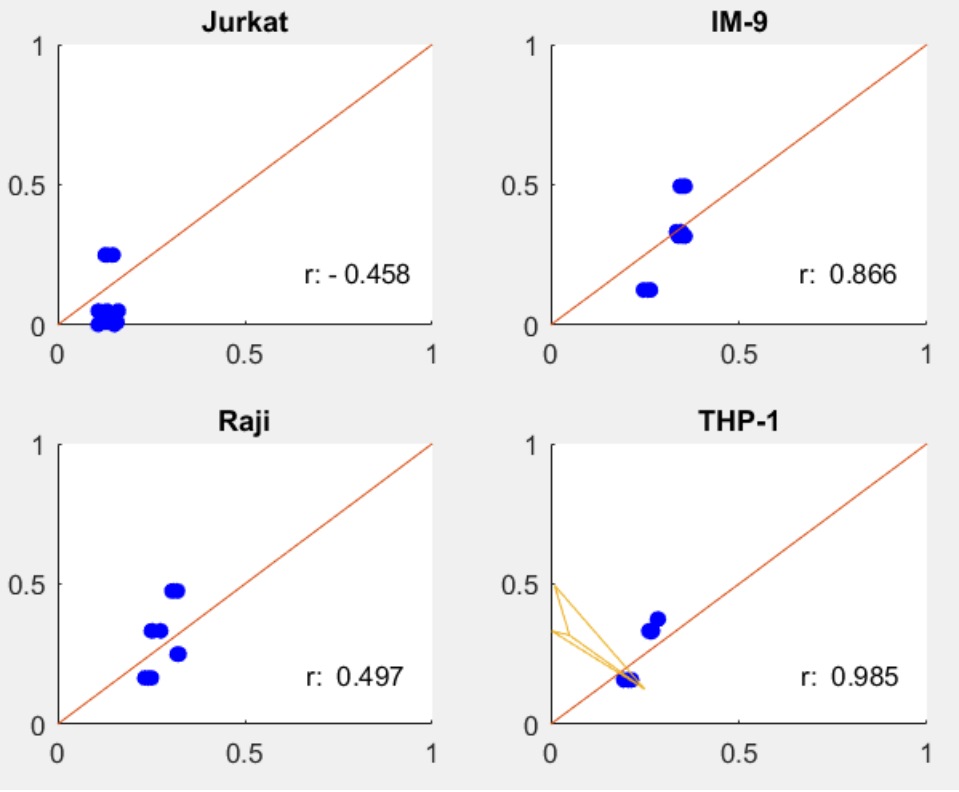

Figure 11. Above are four scatterplots representing the correlation of Deblender clusters to the different tissue types in dataset B.

Deblender struggles with consistently finding good correlation in the B data, both IM-9 and THP-1 do however get good correlation scores.

Dataset B DeCloud Hierarchical clustering

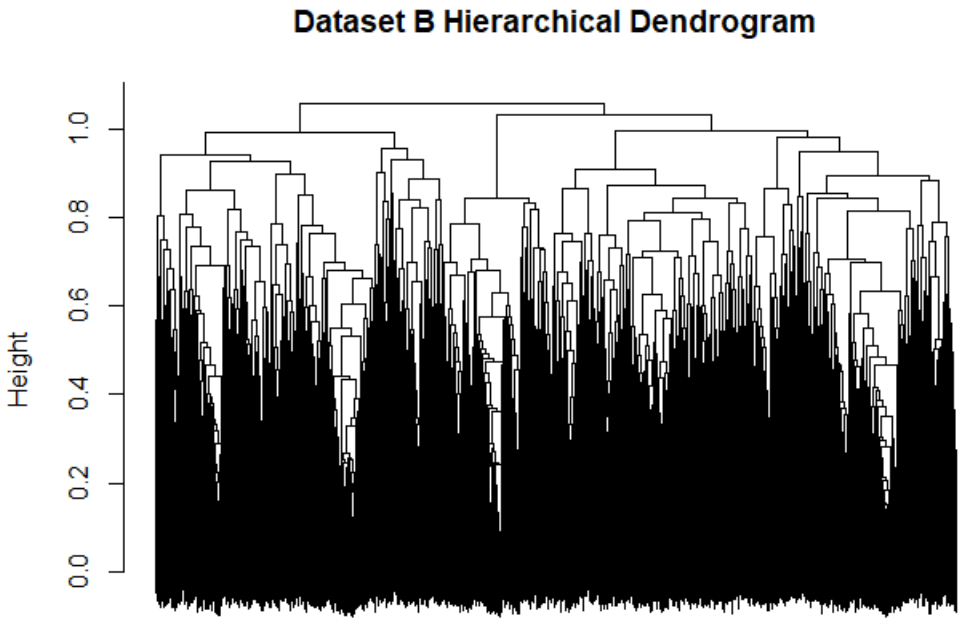

Figure 12. Hierarchical Clustering Dendrogram for test B.

Hierarchical scores well on dataset A, but results seems to suffer when there are lower cluster separation and more noise. As there is no implemented noise filter with the Hierarchical cluster it is not able to perform well on this dataset.

Tool/Algorithm	Jurkat	IM-9	Raji	THP-1
DeCloud/Hierarchical	r : -0.142	r: 0.932	r: 0.630	r: 0.982

Table 8. Represents Pearson correlation score for DeCloud Hierarchical clustering algorithm for each tissue type in dataset B.

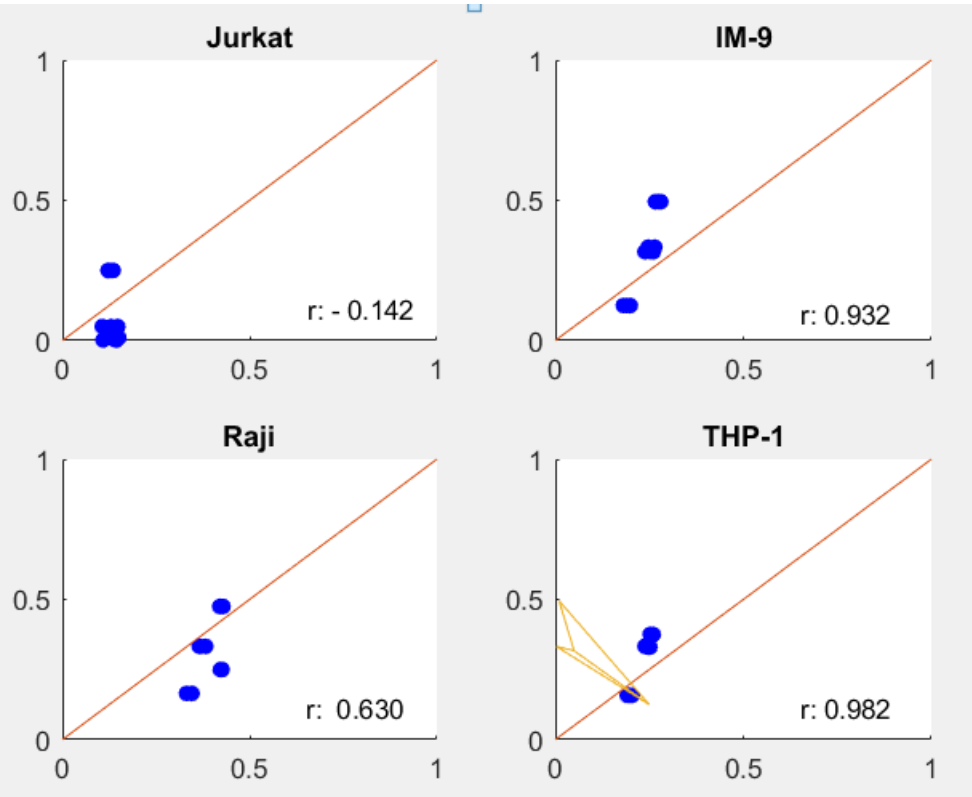

Figure 13. Above are four scatterplots representing the correlation of Hierarchical clusters to the different tissue types in dataset B.

The hierarchical clustering algorithm has similar problems to Deblender and scores poorly on Jurkat with a correlation score of r: -0.142, where 0 means no correlation to the real data. The IM-9 r: 0.932 and THP-1 r: 0.982 are good results, while the Jurkat and Raji does not perform as well.

Dataset B DeCloud PAM

The PAM clustering algorithm without the filter function will suffer similarly to the other clusters as it is not equipped well to handle this type of data set.

Tool/Algorithm	Jurkat	IM-9	Raji	THP-1
DeCloud/PAM	r : - 0.141	r: 0.891	r: 0.556	r: 0.983

Table 9. Represents Pearson correlation score for DeCloud PAM clustering algorithm for each tissue type in dataset B.

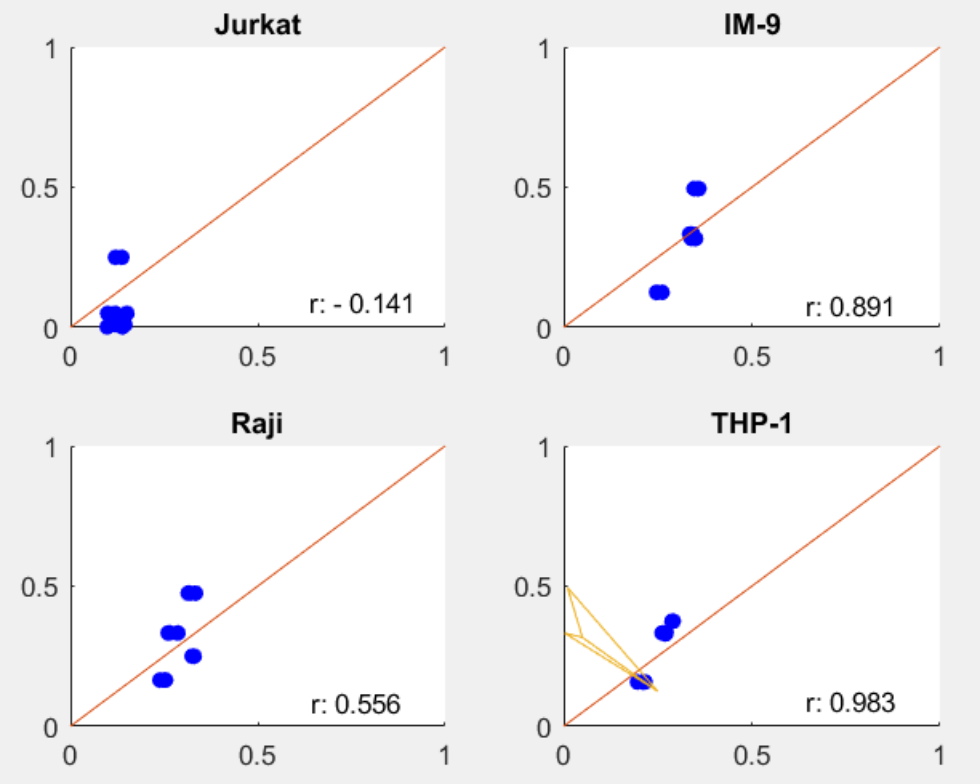

Figure 14. Above are four scatterplots representing the correlation of PAM clusters to the different tissue types in dataset B.

The PAM clustering algorithm without the filtering system implemented does not do as well on all the tissue samples, Jurkat especially did not return very promising result.

These results are not surprising, the data should have been better preprocessed before this step to help the cluster algorithms perform proper classification. The reason why I decided to make this analysis on such difficult data was to show the true benefits of the PAM clustering function with the filtering function implemented.

Dataset B DeCloud PAM with cluster filter

For the PAM clustering algorithm with the cluster filtering system we use the same clustering sets created with the other PAM tests. The filter function is implemented identically to the Dataset A PAM filter, where the top cluster were picked based on the calculated optimal cluster calculation. In this case the optimal cluster calculation equals 4, which means we will find the top four cluster for cluster set 4 to 8. These are the same cluster sets used by the other PAM cluster which scored lowest of all the other clustering algorithms previously.

Cluster Set	1	2	3	4	5	6	7	8
4	**0.159**	**0.164**	**0.159**	**0.188**				
5	0.145	**0.164**	**0.154**	**0.147**	**0.190**			
6	0.141	**0.165**	**0.146**	**0.179**	0.126	**0.167**		
7	**0.156**	**0.174**	**0.194**	**0.168**	0.121	0.110	0.086	
8	**0.156**	0.140	0.117	0.126	**0.193**	**0.187**	**0.175**	0.071

Table 10. Above is a table showing the silhouette width per cluster in cluster sets 4-8. The 4 clusters with the highest silhouette width per cluster set are kept (in bold).

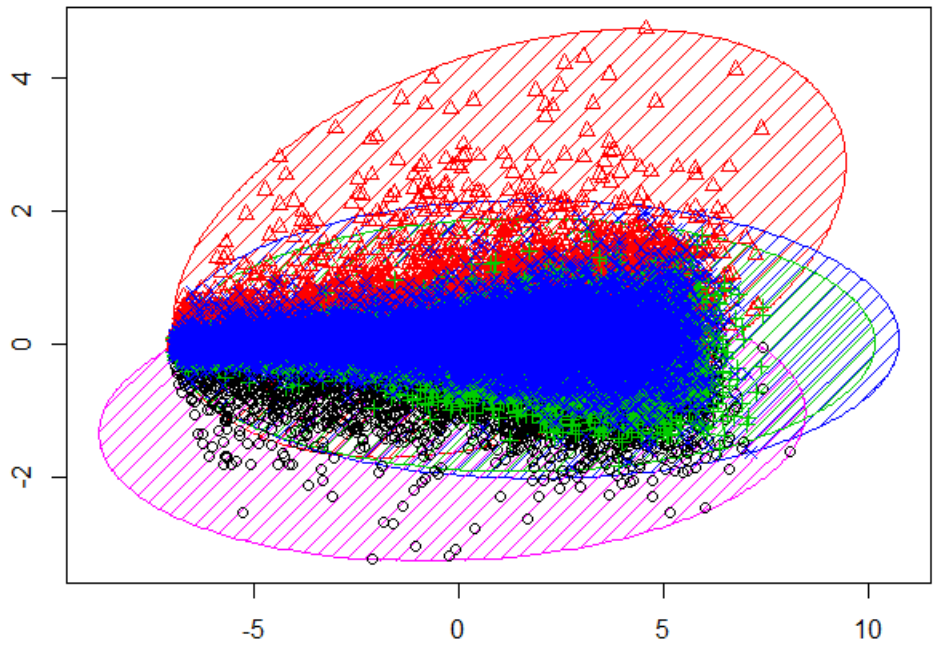

Figure 15. Visual representation of the clusters in cluster set 4 for dataset B.

Figure 15 shows the cluster plot of the PAM cluster set 4 where we can see why the data are so difficult to classify. These are the data the non-filtered PAM clustering algorithm was trying to classify. There much overlapping data and very little separation. The goal of the filtering system is to remove enough unnecessary data from the clusters. This must be done without removing the outlier data with high silhouette confidence which makes each genes unique. Running the Euclidean distance Dunn index we get the score per set (table 11)

Cluster set	Dunn Score
Top 4, 4 clusters	0.00818
Top 4, 5 clusters	0.00854
Top 4, 6 clusters	0.00887
Top 4, 7 clusters	0.00731
Top 4, 8 clusters	0.00836

Table 11. Cluster plot of the set of 4 clusters.

Looking at the Dunn Index scores in table 11 we can see that there is a large drop for cluster set 7, giving a clear minimum. From this it is clear that we should use the top 4 clusters of cluster set 7 for our result.

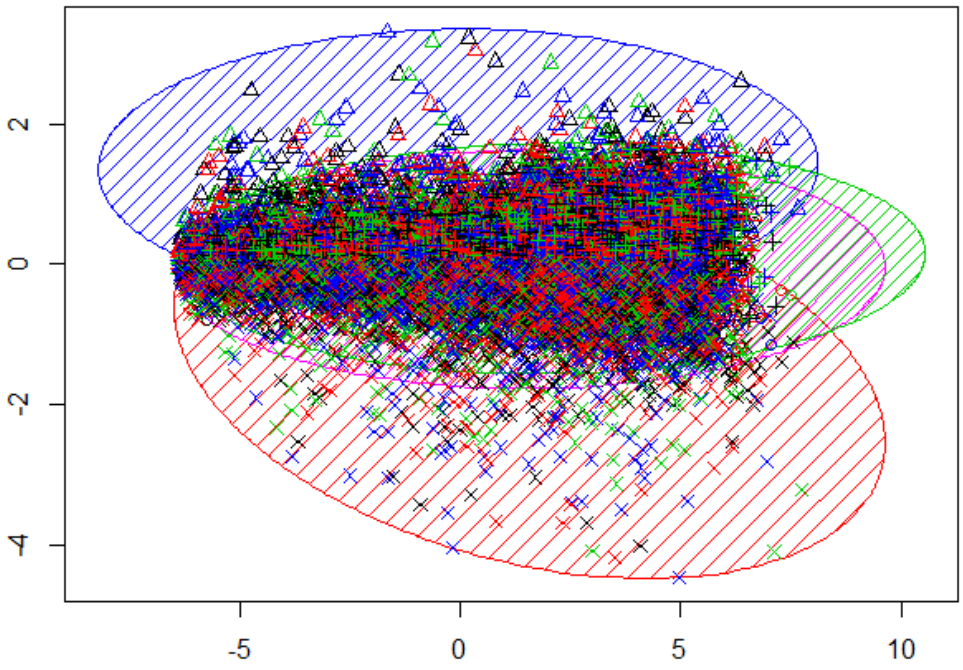

Figure 16. Cluster plot of top 4 clusters in cluster set 7.

In figure 16 we can see the top 4 clusters with much more separation in the data giving the cluster algorithm better opportunity to find the patterns.

Tool/Algorithm	Jurkat	IM-9	Raji	THP-1
DeCloud/PAM with filter	r : 0.981	r: 0.985	r: 0.993	r: 0.989

Table 12. Represents Pearson correlation score for DeCloud PAM clustering algorithm, with cluster filter, for each tissue type in dataset B.

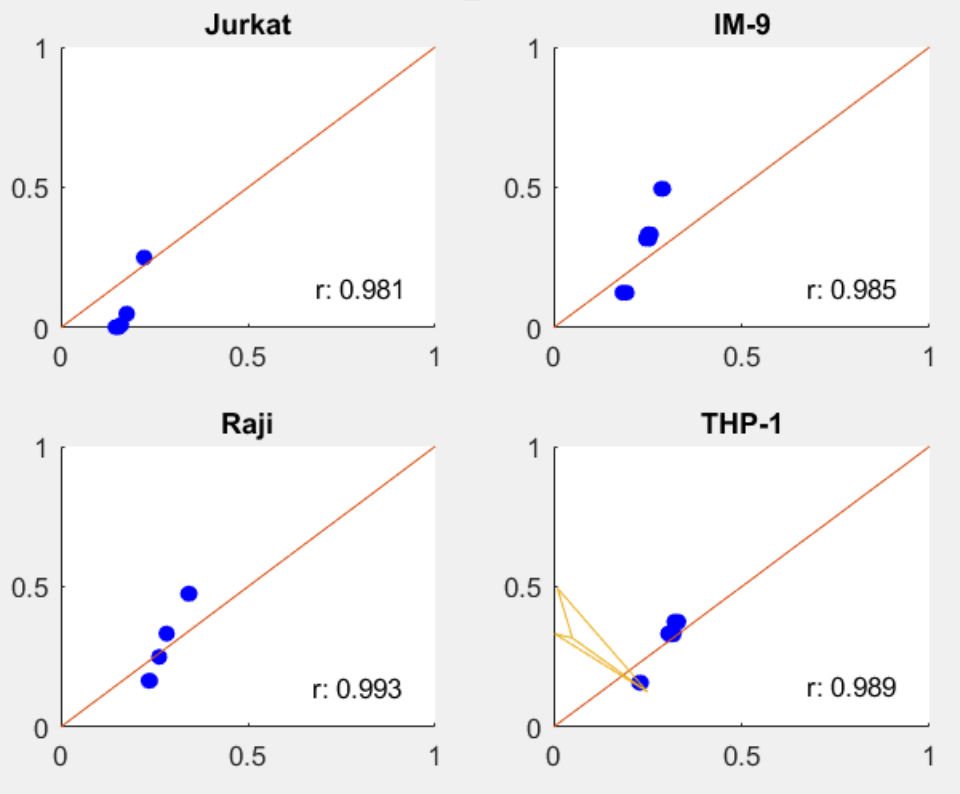

Figure 17. These scatter plots shows the correlation between the estimated proportions and the real values for dataset B.

The filtering system added to the PAM clustering algorithm gives the total Pearson correlation score an increase from r: 0.797 to r: 0.914.This increase is very encouraging and is accomplished without any changes to the data beyond the automated filtering function.

Tool/Algorithm	Pearson total	RMSE
Deblender/K-means	r: 0.786	0.0994
DeCloud/Hierarchical	r: 0.620	0.1158
DeCloud/PAM	r: 0.797	0.0960
DeCloud/PAM (w/cluster filter)	r: 0.914	0.0974

Table 13. Total Pearson correlation for each result on dataset B, with RMSE.

The Pearson total score is created by combining all the individual scores per cell into one total score in order to make the comparison easier per tool/algorithm.

Looking at the total scores for dataset B we can see a big increase in correlation when the filtering process was applied. My observations regarding this data set are that it has a lot of noise which inhibits the standard clustering algorithms without more extensive data preprocessing prior to clustering. By applying the filter function, the system is able to automatically cut through the noise which is limiting the other clustering algorithms from finding the true hidden patterns.

GSE65135 (C)

The GSE65135 data set contains 43668 unknown gene data (rows) and 14 unknown tissue samples (columns). This data set is also very large with a lot of noise in the data. The dataset has 3 cells, B cells, CD4 T cells and CD8 T cells. This dataset does also require us to implement a set number of clusters as the internal validation measures failed to reliably identify how many cells were present in the data. The recommended cluster number is implemented manually for both Deblender and DeCloud. Since the results dataset B saw such an improvement when using the filter system, I chose to include dataset C as Deblender has difficulty clustering it.

Dataset C Deblender

Dataset C is clustered similarly to dataset A and B. The results in Deblender indicate little correlation and do not come close to the same quality of results we saw in the other tests.

Tool/Algorithm	B Cells	CD4 T Cells	CD8 T Cells
Deblender/K-means	r: 0.457	r: - 0.141	r: 0.261

Table 14. Represents Pearson correlation score for Deblender for each tissue type in dataset C.

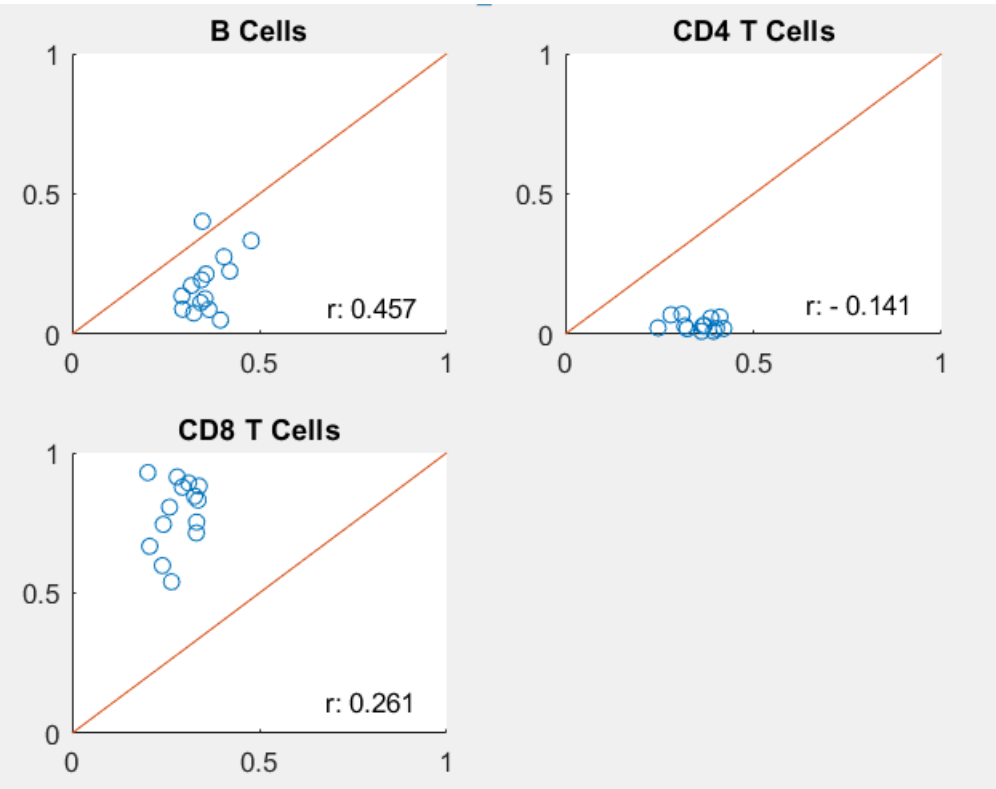

Figure 18. These scatter plots shows the correlation between the estimated proportions and the real values for dataset C.

Dataset C DeCloud Hierarchical

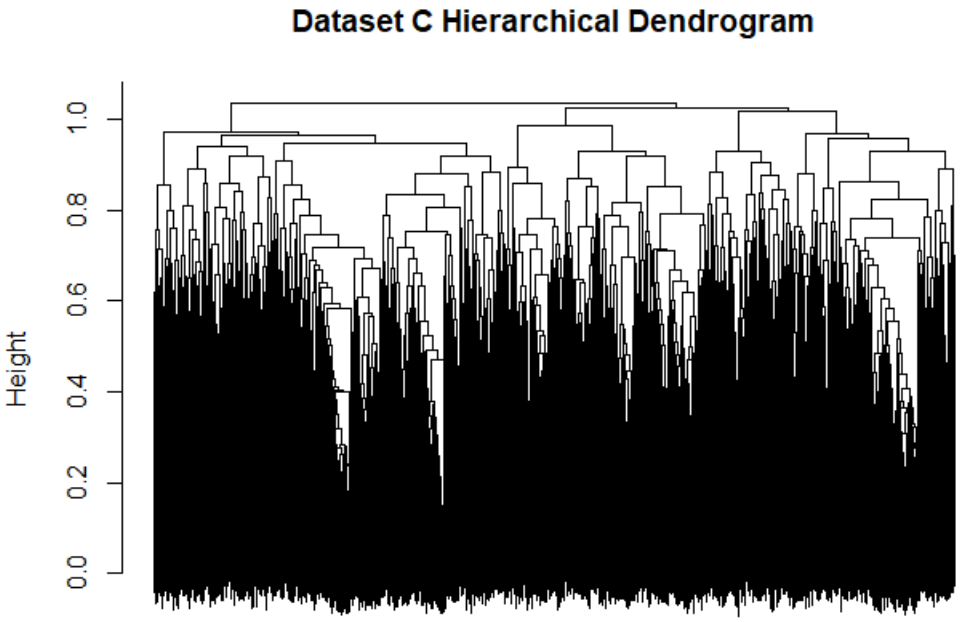

Figure 19. Dendrogram for the hierarchical clustering algorithm on dataset C.

The Hierarchical clustering algorithm was implemented the same as Deblender.

Hierarchical does not perform noticeably better than Deblender does on this dataset does and shows how much difficulty these clustering algorithms have in identifying the gene profiles.

Tool/Algorithm	B Cells	CD4 T Cells	CD8 T Cells
DeCloud/Hierarchical	r: -0.111	r: 0.495	r: 0.311

Table 15.Represents Pearson correlation score for DeCloud PAM clustering algorithm for each tissue type in dataset C.

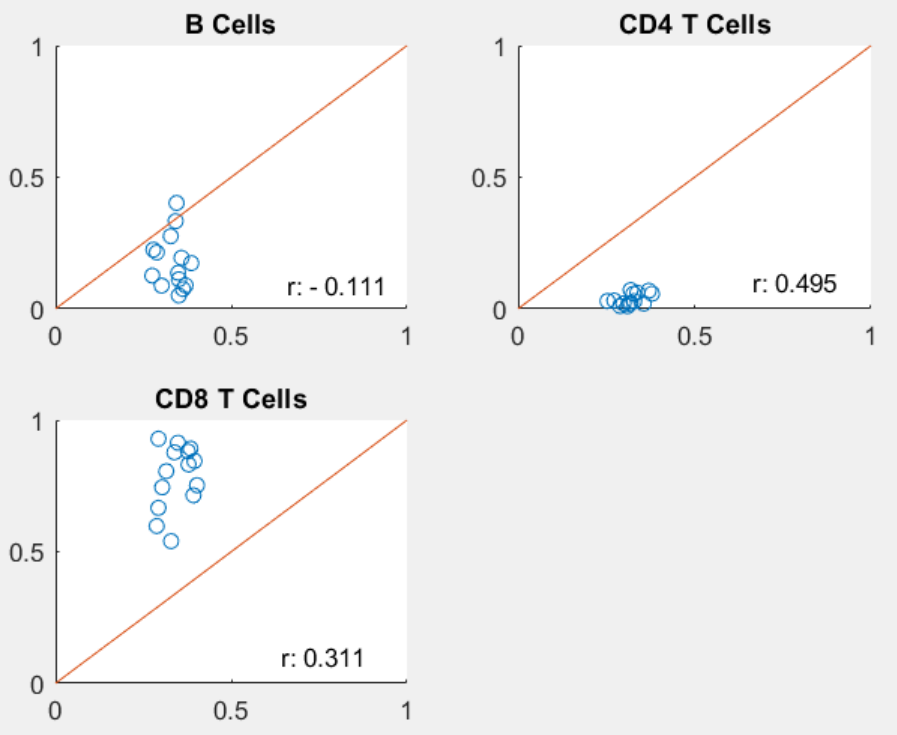

Figure 20. These scatter plots shows the correlation between the estimated proportions and the real values for the C data.

Since a score of 0 means no correlation in the data, B Cells get close to finding no structure in these data on B Cells. Even CD4 T Cells, which have the best result, does not do particularly well compared to our results on the A and B datasets.

Dataset C DeCloud PAM

The PAM clustering algorithm, for this test, had the worst results of all of the clustering algorithms. The clustering algorithm found very little correlation in the data and in the PAM. The average silhouette-width in PAM for the clusters has a very low score in general which indicates low confidence in that the data are in the correct cluster. More on this in the PAM filtered test.

Tool/Algorithm	B Cells	CD4 T Cells	CD8 T Cells
DeCloud/PAM	r: - 0.160	r: - 0.264	r: - 0.419

Table 16. Represents Pearson correlation score for DeCloud PAM clustering algorithm for each tissue type in dataset C.

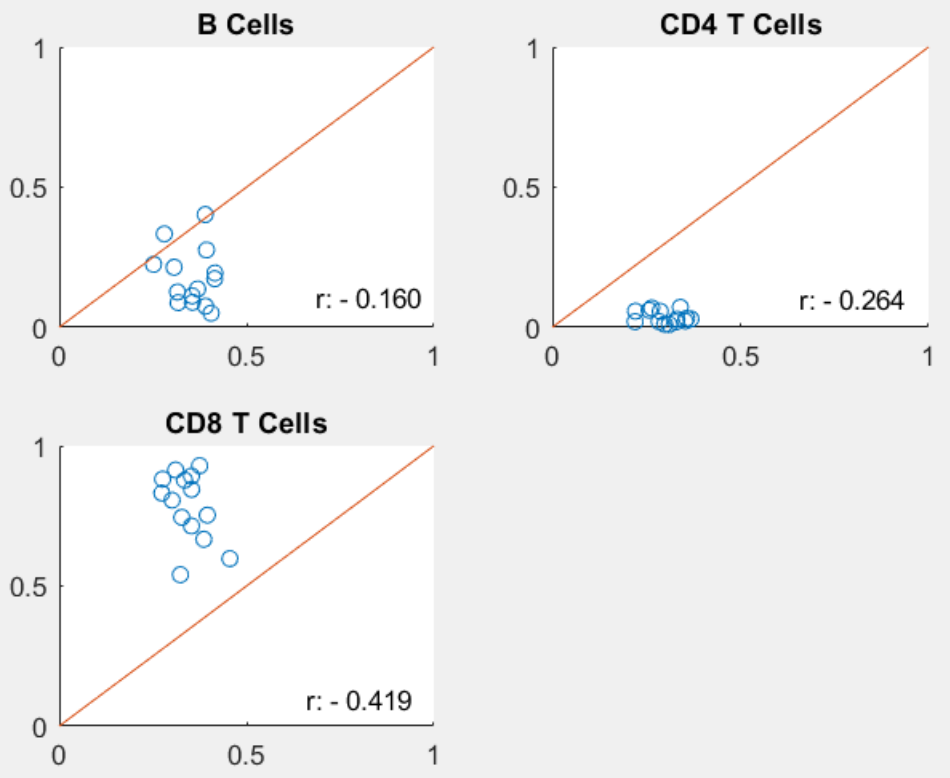

Figure 21. These scatter plots shows the correlation between the estimated proportions and the real values for dataset C.

Dataset C DeCloud PAM with cluster filter

As PAM filter was able to give a big boost to the scores in test B, there were hopes that it could do something similar in this test. Unfortunately, this test has the same results as normal PAM as the clustering algorithm decides that the original cluster set 3 was the best option.

Cluster Set	1	2	3	4	5	6	7	8
3	**0.153**	**0.146**	**0.123**					
4	**0.138**	0.119	**0.156**	**0.120**				
5	**0.159**	**0.163**	**0.135**	0.099	0.123			
6	**0.155**	0.109	**0.157**	**0.149**	0.098	0.121		
7	**0.175**	0.113	0.106	**0.149**	**0.147**	0.092	0.122	
8	**0.154**	0.103	0.075	**0.162**	**0.157**	0.092	0.094	0.121

Table 17. Above is a table showing the silhouette width per cluster in cluster set 3-8. The 3 clusters with the highest silhouette width per cluster set are kept (in bold).

In table 17 we see the average Silhouette-width per cluster in PAM. The scores presented are much lower than for dataset A and a little lower than for dataset B. For filtering, we are looking for the clusters which gives us the most information per set. The most noticeable difference between datasets B and C is the lack of silhouette width improvement between the top cluster set (cluster set 3) and the lower cluster sets (cluster sets 6-8).

Running the Dunn Index(table 18) as we have done in the other tests, we find that the lowest scoring cluster set is cluster set 3, which is the set without filtering. We already know that the result in PAM without the filter is very poor, but as we can see in fig, the confidence in the data is not very high. In this case, the filtering system was unable to improve the score of the normal PAM implementation.

Cluster set	Dunn Score
Top 3, 3 clusters	0.01219
Top 3, 4 clusters	0.01364
Top 3, 5 clusters	0.01355
Top 3, 6 clusters	0.01355
Top 3, 7 clusters	0.01233
Top 3, 8 clusters	0.01356

Table 18. Dunn Index Scores per cluster set.

Tool/Algorithm	B Cells	CD4 T Cells	CD8 T Cells
DeCloud/PAM(w/ cluster filter)	r: - 0.160	r: - 0.264	r: - 0.419

Table 19. Represents Pearson correlation score for DeCloud PAM clustering algorithm, with cluster filter, for each tissue type in dataset C.

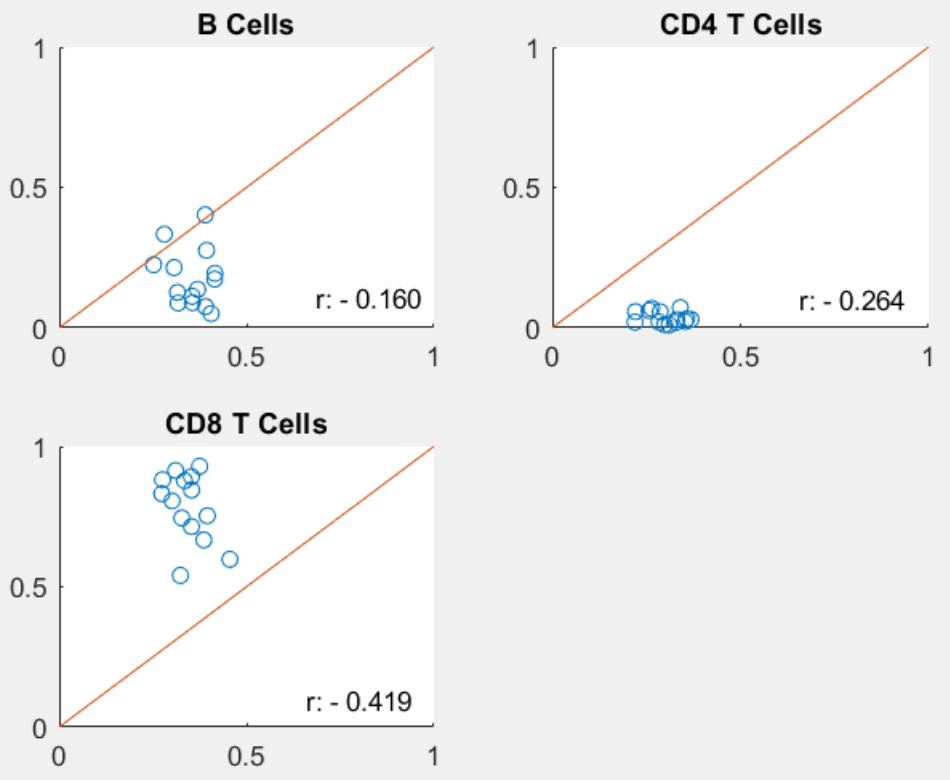

Figure 22. These scatter plots show the correlation between the estimated proportions and the real values for dataset C.

As the filtering system was unable to find any indication that there were better clusters in other cluster sets, it used the normal PAM implementation.

For dataset C we see a case were none of the clustering algorithms were able to properly classify the gene expression profiles. For these data to be properly clustered by any of these clustering algorithms in the future there needs to be better preliminary handling of the data, for our test it provided an opportunity to show how the filtering system functions in a case where it was unable to properly deconvolve the data.

Tool/Algorithm	Pearson total	RMSE
Deblender/K-means	r: 0.394	0.3180
DeCloud/Hierarchical	r: 0.281	0.3282
DeCloud/PAM	r: 0.369	0.3210
DeCloud/PAM (w/cluster filter)	r: 0.369	0.3210

Table 20. Total Pearson correlation for each result on dataset B, with RMSE.

Runtime for results

When analyzing the runtime of Hierarchical clustering and PAM clustering we see a big difference in runtime. The Hierarchical clustering algorithms runs between 5 and 8 times faster than the PAM clustering algorithm on the same datasets.

Clustering Algorithm	Dataset A	Dataset B	Dataset C
Hierarchical clustering	33 min	2 h 8 min	2 h 41 min
PAM clustering	2 h 36 min	16 h 20 min	13 h 6 min

Table 21. The table presents the runtime for each dataset used in the results. PAM clustering represents both PAM with and without the cluster filter function.

The PAM clustering algorithm with filter has a much better score than Hierarchical clustering on both Dataset A and Dataset B in the results but it comes at the cost of time (table 21). This is something that should be considered when choosing between the two clustering algorithms. Another consideration should be that the data sets used (especially dataset B and C) are very large, the smaller the dataset analyzed, the smaller the difference between the clustering algorithms will become. Another way to decrease runtime would be to lower the number of cluster sets calculated.

Discussion

The DeCloud offers a range of new options to continue further exploration into how Deblender can become an even more effective deconvolution tool. The ability to handle count data for RNA sequencing alone is a whole new direction to study, not to mention all the other clustering algorithms not fully explored in this paper. For my purposes, I limited my scope to tests already conducted by Deblender, which were provided to me with the tool and to the clustering algorithms that were able to handle the large datasets. I did do some tests with the RNA sequencing data with reasonable results, but decided against publishing these results due to having to coerce the data from a RPKM format back into a count data format. Although the results from these tests were promising, the lack of proper count data made me choose to go a different direction and focus on Hierarchical and PAM clustering with the filtering function. I do believe that there is a lot of potential in the RNA sequencing clustering algorithms which should be explored given opportunity.

Based on my observations, Hierarchical clustering is the superior algorithm for simple data, which are easily separable due to efficiency. Based on my observations it has a slight edge in accuracy over the K-means clustering algorithm, which is implemented in Deblender. PAM on the other hand offers much more flexibility in regard to how the data are handled, which is why I chose it as the clustering algorithm used for the filtering function.

The part of my paper which I am truly pleased with and which is novel, is the filtering function. Although it does need further testing in order to prove its robustness, it does provide the best clusters and best cluster set for each of the tests I applied it to. When I originally implemented the filtering function I had not expanded my clustering to more than 6 clusters and had not found any good results when analyzing dataset B, but chose to expand my search using my new filtering function. I immediately got the result I am presenting in the results from cluster set 7 and

found that my unorthodox use of the Dunn Index modeled the scoring quality I got from each set quite well. I do not advocate for the filtering process to be implemented blindly however as highly separated clusters with no overlap, especially if dense, will score quite high on the Dunn Index. In such a case, the Dunn Index should be used in the traditional sense, looking for the maximal score. Although this hypothetical situation did not occur in my tests, it would be very important to be aware of this fact in order to get accurate data. To combat this I would suggest implementing a threshold for what the minimum distance there can be between clusters. If the minimum distance is above the given threshold, then the function should look for the maximum Dunn Index score instead of the minimum. Although this theory is untested, it stands to reason that a well-known internal validation function should be able to provide a quality measure as is intended when circumstances are appropriate for it.

Another observation I have made in the data is that clusters with a maximal dissimilarity above 1 will give poor results. The filtering function should be applied in such cases as to either remove such a cluster or to have it ‘split’ in a larger cluster set. In dataset A I found that the removed cluster in cluster set 4 had a maximal dissimilarity above 1. In the case of dataset B, all the initial clusters in the recommended 4 cluster set had maximal dissimilarity over 1. I believe this may also be a way to measure how big the cluster set should be expected to be in order to separate out the data properly. This theory would need much further research in order to be conclusive, but my observation does support it.

When analyzing dataset C there was very little success in terms of deconvolution results; it did however give valuable information regarding the filtering function. For dataset C we saw how the filtering system handled data where it had little confidence in the clusters and low indications of better data in higher cluster sets. Seeing that the function favored not filtering away data in

such a case is very encouraging. I did out of curiosity check the results for the other cluster sets and found that none of them had any improvements over the other clustering algorithms.

The tool as it stands, with the tests I have conducted, does suggest it has major possibilities in terms of increased accuracy in unsupervised deconvolution. The tests are done in comparison to Deblender, a deconvolution tool which is currently pending publication. The results achieved in Deblender compared to other deconvolution tools indicate that it is at the very least comparable to the other deconvolution tools that have been published in the past, this is especially true for the datasets tested in this thesis. Based on my results with the datasets presented in this thesis, there are good arguments for DeCloud being superior for these types of datasets compared to Deblender. Considering the range of new options available in this program, not all of which have been tested, I believe that adding this as an option in Deblender in order to better handle large variations in problems would be very beneficial.

The largest weakness of this implementation is in the runtime of the program and the need for large amount of internal memory. I would highly recommend running this program on a server cluster if the datasets are larger than a few thousand rows, as I have done.

As a closing statement, I feel that DeCloud has proven to be a successful deconvolution tool with good enough results to warrant consideration for future use and further testing.

References

1. Phillips, T. Regulation of transcription and gene expression in eukaryotes. *Nature Education*, *1*(1):199, 2008.
2. What is gene expression. Yourgenome. Visited 12/04/2018. (https://www.yourgenome.org/facts/what-is-gene-expression)
3. Can genes be turned on and off in cells?. How genes work. Visited 12/04/2018 (https://ghr.nlm.nih.gov/primer/howgeneswork/geneonoff)
4. Essentials of Cell Biology. Ch 2.3 Differential Control of Transcription and Translation Underlies Changes in Cell Funtion. Visited 12/04/2018. (https://www.nature.com/scitable/ebooks/essentials-of-cell-biology-14749010/122996928)
5. Nature Definition MicroArray. Visited 12/04/2018. (http://www.nature.com/scitable/definition/microarray-202)
6. Brazma A, Vilo J. Gene Expression Data Analysis. FEBSPress. 480(1):17-24. 2000.
7. Wang, Z., Gerstein, M., Snyder, M. RNA-Seq: a revolutionary tool for transcriptomics. Nat Rev Genet 10 (1) 57-63, 2009.
8. Keren H, Lev-Maor G, Ast G. Alternative splicing and evolution: diversification, exon definition and function. Nature Reviews Genetics volume 11, pages 345–355,2010.
9. Ozsolak F, Milos P. RNA sequencing: advances, challenges and opportunities. Nat Rev Genet. 12(2): 87–98, 2011.

10. Liebner D, Huang K, Parvin J. MMAD: microarray microdissection with analysis of differences is a computational tool for deconvoluting cell type-specific contributions from tissue samples. Bioinformatics 30(5):682-9, 2014

11. Gaujoux R, Seoighe C. CellMix: a comprehensive toolbox for gene expression deconvolution. Bioinformatics, Vol 29, Issue 17, Pages 2211–2212, 2013.

12. Hartmann N, Kohane I, Brinkmann V, Staedtler F, Letzkus M, Bongiovanni S, Szustakowski J. Optimal Deconvolution of Transcriptional Profiling Data Using Quadratic Programming with Application to Complex Clinical Blood Samples. PLoS ONE 6(11), 2011.

13. Konstantina Dimitrakopoulou; Elisabeth Wiik; Lars A. Akslen; Inge Jonassen. Deblender: a semi-/unsupervised multi-operational computational method for complete deconvolution of expression data from heterogeneous samples. BMC Bioinformatics, submitted.

14. Matlab cluster analysis. Mathworks. Last Visited 12/04/2018. (https://www.mathworks.com/discovery/cluster-analysis.html).

15. Matlab unsupervised learning. Mathworks. Last Visited 12/04/2018. (https://www.mathworks.com/discovery/unsupervised-learning.html).

16. Matlab pattern recognition. Mathworks. Last Visited 12/04/2018. (https://www.mathworks.com/discovery/pattern-recognition.html).

17. R partitioning around medoids. R documentation. Last Visited 12/04/2018. (https://stat.ethz.ch/R-manual/R-devel/library/cluster/html/pam.html).

18. Sekula M. optCluster: an R package for determining the optimal clustering algorithm and optimal number of clusters. The University of Louisville's Institutional Repository. 2015

19. Matlab K-means function. Mathworks. Last Visited 12/04/2018. (http://www.mathworks.com/help/stats/K-means.html).
20. R clara. R documentation. Last Visited 12/04/2018. (https://www.rdocumentation.org/packages/cluster/versions/2.0.6/topics/clara).
21. R Self-Organized Tree Algorithm(SOTA). R documentation. Last Visited 12/04/2018. (https://www.rdocumentation.org/packages/clValid/versions/0.6-6/topics/sota).
22. Matlab negative binomial distribution. Mathworks. Last Visited 12/04/2018. (https://www.mathworks.com/help/stats/negative-binomial-distribution.html)
23. Matlab poisson distribution. Mathworks. Last Visited 12/04/2018. (https://www.mathworks.com/help/stats/poisson-distribution.html).
24. Mitchell T. Machine Learning. McGraw-Hill Science/Engineering/Math: 191 – 196, 1997.
25. Parekh P, Katselis D, Beck C, Salapaka S. Deterministic Annealing for Clustering: Tutorial and Computational Aspects. American Control Conference. 2015
26. Yasuda M. Deterministic Annealing: A Variant of Simulated Annealing and its Application to Fuzzy Clustering. National Institute of technology Japan. 2017.
27. Carr, R. Simulated Annealing. MathWorld. Last Visited 12/04/2018. (http://mathworld.wolfram.com/SimulatedAnnealing.html)
28. Kim S, Pena M, Moll M, Giannakopoulos G, Bennet G, Kavraki L. An evolution of different clustering methods and distance measures used for grouping metabolic pathways. Bioinformatics and computational biology. 2016.
29. R package ‘optCluster’. R documentation. Last Visited 12/04/2018. (https://cran.r-project.org/web/packages/optCluster/optCluster.pdf).

30. MedlinePlus ‘Genes’. Medical Encyclopedia. Last Visited 12/04/2018. (https://medlineplus.gov/ency/article/002371.htm).

31. MedlinePlus ‘Protein in diet’. Medical Encyclopedia. Last Visited 12/04/2018. (https://medlineplus.gov/ency/article/002467.htm).

32. Johnson A, J Lewis. Molecular Biology of the cell. 4th edition. Garland Science. 2002.

33. Ozsalak F, Milos P. RNA sequencing: advances, challenges and opportunities. Nat Rev Genet. 87-98, 2011.

34. Zoghbi H, Beaudet A. Epigenetics and Human Disease. Cold spring Harbor Perspective in Biology. 2016.

35. BRCA Facts ’BRCA Mutations: Cancer Risk and Genetics testing’. Cancer.gov. Last Visited 12/04/2018. (https://www.cancer.gov/about-cancer/causes-prevention/genetics/brca-fact-sheet#q1)

36. Hernandez L, Blazer D. Genes, behavior, and the social environment: Moving beyond the nature/nurture debate. ’Genetic Health’. National Academies Press; 2006.

37. Biomarkers in risk assessment: Validity and Validation. WHO report. 2001. Visited 12/04/2018. (http://www.inchem.org/documents/ehc/ehc/ehc222.htm)

38. Sweeney T, Khatri P. Generalizable Biomarkers in Critical Care: Towards Precision Medicine. Crit Care Med. 45(6): 934-939, 2017

39. Prasetyanti P, Medema J. Intra-Tumor heterogeneity from a cancer stem cell perspective. Mol Cancer 16: 41, 2017.

40. Ruysscher D, Jin J, Lautenschlaeger T, She J, Liao Z, Kong F. Blood-based biomarkers for precision medicine in lung cancer: precision riation therapy. Translational Lung Cancer Research 661-669, 2017.

41. Zhong, Y, Wan Y, Pang K, Chow L, Liu Z. Digital Sorting of Complex Tissues for cell-specific gene expression profiles. BMC Bioinformatics 14: 89, 2013.

42. Storey J, Madeoy J, Strout J, Wurfel M, James R, Joshua Akey. Gene Expression Variation within and among human populations. Am J Hum Genet 80(3): 502–509, 2007.

43. Clancy S. RNA Functions. Nature Education 1(1):102, 2008.

44. Alberts B, Johnson A, Lewis J. An Overview of Gene Control. Garland Science; 2002.

45. Mohammedi S, Zuckerman N, Goldsmith A, Grama A. Acritical Survey of Deconvolution Methods for Separating cell-types in Complex Tissues. Foundations and Applications of Science of Information. 2015.

46. Gaujoux R. CellMix: A comprehensive toolbox for gene expression deconvolution. Bioinformatics*, Volume 29, Issue 17, Pages 2211–2212, 2013.*

47. Nguyen T, Bhatti A, Yang S, Nahavandi S. RNA-Seq Count Data Modelling by Grey Relational Analysis and Nonparametric Gaussian Process. PLOS One. 2016.

48. R package 'RankAggreg'. R documentation. Last Visited 12/04/2018. (https://cran.r-project.org/web/packages/RankAggreg/vignettes/RankAggreg.pdf)

49. Tarca A, Romero R, Draghici S. Analysis of microarray experiments of gene expression profiling. Am J Obstet Gynecol. 195(2): 373-388, 2006.

50. Sedgwick, P. Pearson's correlation coefficient. BMJ, 345, 2012.

Appended A. DeCloud code:

```
library(MBCluster.Seq)

library(optCluster)

library(combinat)

library(readr)

library(BiocGenerics)

library(RankAggreg)

library(readr)

library(cluster)

library(clValid)

library(caret)

library(doParallel)

library(matlabr)

library(R.matlab)

library(dplyr)

library(ggplot2)

library(data.table)

library(Biobase)

library(pacman)

library(annotate)

library(mclust)

library(fpc)

library(affy)

library(EBSeq)



#To import the data, run the code below. Uncomment the GSE11058(B) or GSE65135(C) to import those datasets instead.

#GSE19830(A)----------------------------------------------------------

CData = as.data.frame(readMat('clust_high_variable_data.mat'))

Gene = as.data.frame(readMat('clust_high_variable_gene.mat'))
```

```
HVData = as.data.frame(readMat('high_variable_data.mat'))
Gene = t(data.frame(Gene))
row.names(CData) = c(Gene)
row.names(HVData) = c(Gene)
#------------------------------------------------------------

#GSE11058(B)----------------------------------------------------
#CData = as.data.frame(readMat('GSE11058_Clust_HVD.mat'))
#Gene = as.data.frame(readMat('GSE11058_HVG.mat'))
#HVData = as.data.frame(readMat('GSE11058_HVD.mat'))
#Gene = t(data.frame(Gene))
#row.names(CData) = c(Gene)
#row.names(HVData) = c(Gene)
#------------------------------------------------------------

#GSE65135(C)----------------------------------------------------
#CData = as.data.frame(readMat('GSE65135_cluster_data.mat'))
#Gene = as.data.frame(readMat('GSE65135_gene.mat'))
#HVData = as.data.frame(readMat('GSE65135_data.mat'))
#Gene = t(data.frame(GSE65135_gene))
#row.names(CData) = c(Gene)
#row.names(HVData) = c(Gene)
#------------------------------------------------------------

#Creates a list of cluster algorithms, in this case only PAM and Hierarchical is being called.
algo = list()
algo = c("Pam", "Hierarchical")

#Number of clusters created, this will create the clustersets 2-8.
l = 2
```

```
u = 8


#Calling the clustering algorithm, algo[1] calls the first word in the list(Pam), algo[2] calls Hierarchcical.

#RankMethod can be changed to CE for cross-entropy and distance can be changed from Spearman to Kendall.

deClusters=  optCluster(CData,l:u,clMethods=c(algo[1],algo[2]),countData = FALSE,validation = "internal",hierMethod = "average",annotation = "null",clVerbose = TRUE, rankMethod = "GA", distance = "Spearman",metric = "correlation", importance = c(1,1,1))



#To call Pam function after clustering is finished. This will create the csv file that can be transported into Deblender. This function also holds the filtering system.

if("Pam" %in% algo){

  do.call(Pam, c(deClusters,l,u))

}

#To call Hierarchical function after clustering is finished. Same as Pam function, just without the clustering function.

if("Hierarchical" %in% algo){

  do.call(Hierarchical(deClusters))

}



Pam = function(deClusters, l, u){

  DunnCluster = list()

  ColSumTable =list()

  #oc will use the calculated optimal cluster set in rankAggreg for test A.

  oc = deClusters@rankAgg$top.list

  oc = sapply(strsplit(oc, "-"), "[", 2)

  oc = as.numeric(oc[1])

  #if running test B or C, please comment out the three lines of code above(oc lines) and uncomment the oc = 3 below.
```

```
#oc = 3
toc=(u-l)+1
loc = (oc-l)+1
for(j in loc:toc){
 silAvg = as.data.table(deClusters@clVal@clusterObjs$pam[[j]]$silinfo$clus.avg.widths)
 silAvg$cluster =  rownames(silAvg)
 topSilC=head(order(-silAvg$V1),oc)
 #Cluster holds the designated cluster number per row.
 cluster = deClusters@clVal@clusterObjs$pam[[j]]$cluster
 # CData.clv holds the data clustered on.
 CData.clv =as.data.frame(cbind(cluster, CData))
 # HVData.clv holds output data values.
 HVData.clv =as.data.frame(cbind(cluster, HVData))

 #Creating a table with data the expression data, adding cluster number on first row.
 for(k in 1:oc){

  CData.clvO = subset(CData.clv, CData.clv$cluster == topSilC[[k]])
  if(k == 1){
   CData.clvc = CData.clvO
  }
  else{
  CData.clvc = rbind(CData.clvc, CData.clvO)
  }
 }
 for(k in 1:oc){

  HVData.clvO = subset(HVData.clv, HVData.clv$cluster == topSilC[[k]])
  if(k == 1){
   HVData.clvc = HVData.clvO
```

```
  }
  else{
  HVData.clvc = rbind(HVData.clvc, HVData.clvO)
  }
 }
 #calculating Dunn Index per cluster set.
 DunnCluster[[j]] = dunn(Data = CData.clvc[,-1], clusters = CData.clvc$cluster, method = "euclidean")

 clvcD = HVData.clvc[,-1]
 clvcD =  split(clvcD, f= CData.clvc[,1])
 sCd = as.data.frame(colMeans(clvcD[[1]]))
 #Creating a column summerized table
 for(i in 2:oc){
  sClust[[i]] = as.data.frame(colMeans(clvcD[[i]]))
  sCd = as.data.frame(cbind(sCd, sClust[[i]]))
 }

ColSumTable[[j]] = t(sCd)



}

DunnCluster = unlist(DunnCluster)
#finding the minimum dunn value cluster set
dunnClust = which.min(DunnCluster)
dunnPos = ((dunnClust-1)+loc)

#print out cluster results from PAM.
write.table(ColSumTable[[loc]],file="PAM.csv",sep="\t", col.names = F, row.names = F)
#print out cluster results from PAM with filter.
```

```
  write.table(ColSumTable[[dunnPos]],file="PAMwFilter.csv",sep="\t", col.names = F, row.names = F)
}



Hierarchical = function(deClusters){
  #comment out oc and uncomment oc = 3 if data set C is being run, change oc to 4 if dataset B is run.
  oc = deClusters@rankAgg$top.list
  oc = sapply(strsplit(oc, "-"), "[", 2)
  oc = as.numeric(oc[1])
  #oc = 3
  #Cut dendrogram to get clusters.
  cluster  = cutree(deClusters@clVal@clusterObjs$hierarchical, k = oc)
  clv =cbind(cluster, HVData)

  for(k in 1:oc){

    clvO = subset(clv, clv$cluster == k)
    if(k == 1){
      clvc = clvO
    }
    else{
    clvc = rbind(clvc, clvO)
    }
  }
  clvcD = clvc[,-1]
  clvcD =  split(clvcD, f= clvc[,1])
  sCd = colMeans(clvcD[[1]])

  for(i in 2:oc){
```

```
   sClust[[i]] = as.data.frame(colMeans(clvcD[[i]]))
   sCd = as.data.frame(cbind(sCd, sClust[[i]]))
  }

  ColSumTable = t(sCd)
  write.table(ColSumTable,file="Hierarchical.csv",sep="\t", col.names = F, row.names = F)

}
```

Appended B. Using Deblender

Deblender is available here: https://github.com/kondim1983/Deblender.

A zipfile is added with the thesis submission, it holds the test results for the clustering. It also has the deblender files which are changed inorder to implement results. There are comments in the matlab code to guide the process. Only file that needs to change is calc_A_unsupervised, the code is already built in and just needs to be uncommented depending on which test needs to run. Deblender's clustering algorithm is also commented out, it will need uncommented for deblender's test to run.

Inge will recieve the datasets and clusters needed if there is a need to rerun my decloud application. The file size for this data is too large to be added next to the thesis for the submission.